I PENDAHULUAN

1.1 Latar Belakang

Permasalahan mengidentifikasi kelas dan lokasi objek pada citra disebut dengan permasalahan *object detection* atau deteksi objek (Sultana *et al* 2019). *Object detection* dapat dilakukan dengan berbagai teknik seperti *frame differencing*, *optical flow*, dan *background subtraction* (Panchal *et al*. 2015). Setiap teknik memiliki kelebihan dan kekurangan masing-masing dalam aspek akurasi maupun waktu komputasi (Prajapati dan Galiyawala 2015). Dewasa ini, metode tradisional berbasis fitur yang diekstraksi secara *hand-crafted* tersebut telah tergantikan dengan metode *deep learning* berbasis *convolutional neural networks* (CNNs) (Trigueros *et al*. 2018). Dengan ketersediaan data dalam jumlah besar serta spesifikasi perangkat keras yang mendukung, *object detection* menggunakan pendekatan *deep learning* telah meraih pencapaian yang lebih baik secara signifikan daripada metode tradisional dalam hal akurasi dan presisi (Trigueros *et al*. 2018).

Beberapa penelitian yang mengangkat permasalahan *object detection* menggunakan pendekatan *deep learning* telah dilakukan termasuk dalam bidang pertanian. Ramcharan *et al*. (2019) menggunakan model Single Shot Multibox Detector (SSD) dengan detektor MobileNet untuk mendeteksi penyakit pada singkong melalui citra daun. Arsenovic *et al*. (2019) melakukan eksperimen terhadap berbagai model *object detection* untuk mendeteksi penyakit tanaman. Secara umum, hasil dari percobaan menunjukkan bahwa metode *two-stage*, dalam penelitian tersebut adalah Faster Region-based CNN (Faster R-CNN), memiliki kinerja yang lebih baik daripada model dengan metode *one-stage*, yaitu YOLOv3, SSD dan RetinaNet. Akan tetapi, metode *two-stage* tersebut secara signifikan lebih lambat dalam hal komputasi (Arsenovic *et al*. 2019). Masih terkait dengan bidang pertanian, Ampatzidis *et al*. (2017) membahas aplikasi dan manajemen robotik untuk tanaman dan penyakit tanaman dari berbagai aspek termasuk *object detection* dengan *deep learning*. Selain itu, Sa *et al*. (2016) menerapkan model Faster R-CNN untuk mendeteksi buah-buahan dan memperoleh skor F1 (*F1 score*) 0,838.

Penelitian terkait implementasi *object detection* berbasis *deep learning* pada perangkat komputasi terbatas di antaranya dilakukan oleh Zhong *et al*. (2018). Penelitian tersebut bertujuan menghitung dan mengenali serangga pada *greenhouse* menggunakan metode *object detection You Only Look Once* (YOLO) dan metode klasifikasi *Support Vector Machines* (SVM). Sistem pengenalan dan penghitungan tersebut kemudian diimplementasikan pada Raspberry Pi. Percobaan yang dilakukan terhadap enam spesies serangga menghasilkan nilai rata-rata akurasi penghitungan serangga 92,50%, rata-rata akurasi klasifikasi 90,18%, dan siklus deteksi dan pengenalan membutuhkan waktu sekitar lima menit pada sistem Raspberry Pi (Zhong *et al*. 2018).

Meskipun memiliki akurasi yang lebih baik daripada metode tradisional, metode *deep learning* menghasilkan waktu komputasi dan pemakaian memori yang lebih tinggi. Hal ini menjadi tantangan ketika ingin menerapkan algoritme berbasis *deep learning* pada perangkat dengan kemampuan komputasi yang terbatas seperti Raspberry Pi. Perangkat seperti ini memiliki batasan dalam hal sumber daya perangkat keras (*hardware*), biasanya terdiri dari prosesor *single-core* dan jumlah

Random Access Memory (RAM) yang terbatas (Foley dan O'Reilly 2018). Dengan harganya yang murah dan ukurannya yang kecil, Raspberry Pi lebih disukai untuk penggunaan dalam kasus-kasus tertentu seperti pada sistem pengawasan dan *Internet of Things* (IoT).

Aspek akurasi dan efisiensi komputasi menjadi *trade-off* dalam implementasi algoritme *object detection* berbasis *deep learning* pada perangkat komputasi terbatas seperti Raspberry Pi. Akurasi yang tinggi dengan waktu komputasi yang dapat ditoleransi adalah kondisi yang ideal untuk dicapai dalam berbagai kasus atau permasalahan yang dihadapi. Oleh karena itu, penelitian ini bertujuan untuk melakukan analisis perbandingan terhadap kinerja beberapa algoritme *object detection* berbasis *deep learning* yang diimplementasikan pada perangkat komputasi terbatas yang mana dalam kasus ini adalah Raspberry Pi.

Melon (*Cucumis melo* L. atau kemudian disingkat *C. melo*) merupakan salah satu spesies tanaman yang sangat sensitif terhadap hama dan penyakit (Balliu dan Sallaku 2017). Mengetahui gejala penyakit yang timbul pada tanaman terinfeksi sangat penting sebagai langkah awal dalam mengidentifikasi penyakit tanaman (Seminis). Identifikasi penyakit tanaman dengan benar pada saat kemunculan pertama kali merupakan langkah yang sangat penting sebagai upaya pengelolaan penyakit tanaman secara efisien (Mohanty *et al*. 2016). Salah satu cara untuk mengidentifikasi penyakit tanaman adalah dengan teknik *digital image processing* atau pengolahan citra digital (Khirade dan Patil 2015). Keberadaan penyakit atau kelainan di suatu tanaman dapat ditandai dengan adanya perubahan warna daun atau pola tertentu seperti bercak yang menunjukkan adanya penyakit pada tanaman tersebut. Kelainan yang sering terjadi pada melon disebabkan oleh hama atau penyakit karena tanaman ini sangat sensitif terhadap keduanya, tetapi dapat pula disebabkan faktor lain seperti kekurangan nutrisi.

Penelitian tetang identifikasi penyakit pada tanaman melon berbasis pengolahan citra di antaranya dilakukan oleh Pineda *et al*. (2018). Penelitian tersebut mengidentifikasi penyakit tanaman melon yang disebabkan oleh bakteri *Dickeya dadantii* menggunakan metode klasifikasi (Pineda *et al*. 2108). Bagaimanapun, klasifikasi dan *object detection* adalah dua permasalahan yang berbeda. Selain itu, belum terdapat penelitian mengenai deteksi penyakit tanaman melon menggunakan algoritme *object detection* berbasis *deep learning* terlebih dengan implementasi pada perangkat dengan kemampuan komputasi yang terbatas. Oleh karena itu, studi kasus yang dipilih pada penelitian ini adalah deteksi penyakit pada daun tanaman melon menggunakan metode *object detection* berbasis *deep learning*.

Penelitian ini merupakan salah satu rangkaian riset pengembangan robot pengawasan (*surveillance robot*) pada tanaman melon. Robot ini diharapkan dapat mendukung pertanian 4.0 yaitu dengan melakukan otomasi dalam mendeteksi kelainan pada tanaman seperti penyakit atau defisiensi nutrisi. Robot tersebut kemudian akan melakukan tindak lanjut untuk mengatasi kelainan tersebut seperti *pruning* pada daun abnormal, mengirim sinyal sebagai peringatan dini, dan sebagainya. Deteksi kelainan pada daun tanaman yang dilakukan oleh robot mengandalkan citra yang diambil melalui kamera yang dipasang pada robot, sehingga penerapan metode *object detection* diperlukan untuk pengembangan robot. Adapun kontribusi penelitian ini adalah menyarankan algoritme *object*

detection berbasis *deep learning* yang lebih optimal untuk ditanamkan pada robot pengawasan tersebut berdasarkan aspek akurasi dan biaya komputasi.

1.2 Perumusan Masalah

Permasalahan yang dihadapi dalam penelitian ini adalah:

a. Bagaimana mengimplementasikan algoritme *object detection* berbasis *deep learning* pada perangkat komputasi terbatas untuk mendeteksi kelainan pada daun tanaman melon?
b. Bagaimana kinerja metode *object detection* berbasis *deep learning* yang unggul dalam hal akurasi namun memiliki waktu komputasi yang efisien pada perangkat komputasi terbatas?

1.3 Tujuan Penelitian

Adapun tujuan dari penelitian ini meliputi:

a. Melakukan evaluasi terhadap model *object detection* berbasis *deep learning* dalam mendeteksi kelainan daun tanaman melon.
b. Mengimplementasikan algoritme-algoritme *object detection* berbasis *deep learning* pada Raspberry Pi untuk mendeteksi kelainan pada daun tanaman melon.
c. Melakukan analisis perbandingan terhadap kinerja algoritme-algoritme *object detection* berbasis *deep learning* dalam implementasinya pada perangkat komputasi terbatas.
d. Menentukan algoritme *object detection* terbaik untuk diterapkan pada perangkat komputasi terbatas dalam mendeteksi kelainan daun tanaman melon.

1.4 Manfaat Penelitian

Penelitian ini diharapkan memberikan beberapa manfaat lebih lanjut seperti:

a. Meningkatkan efektivitas dan efisiensi dalam pemberian kontrol terhadap kelainan tanaman khususnya tanaman melon.
b. Menjelaskan tantangan dan menawarkan solusi dalam implementasi algoritme *object detection* berbasis *deep learning* pada perangkat komputasi terbatas.
c. Mendukung rencana penelitian selanjutnya mengenai robot pengawas pada *greenhouse*.

1.5 Ruang Lingkup

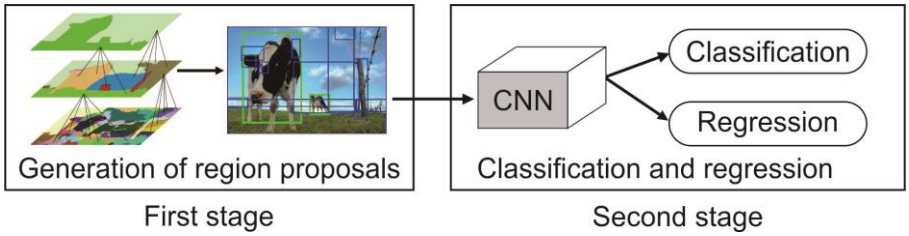
Adapun ruang lingkup penelitian ini mencakup:

a. Data yang digunakan yaitu citra sekumpulan daun melon yang diambil dari iSurf Lab (*IoT for Smart Urban Farming Laboratory*) departemen Ilmu Komputer FMIPA IPB dan *greenhouse Agribusiness and Technology Park* (ATP) IPB Cikarawang.
b. Metode yang dibandingkan adalah Faster R-CNN, SSD, dan YOLOv3.
c. Perangkat komputasi terbatas yang digunakan adalah Raspberry Pi 4.

II TINJAUAN PUSTAKA

2.1 *Object Detection* dengan *Deep Learning*

Dalam *computer vision*, permasalahan *object detection* merupakan permasalahan mengestimasi kelas dan lokasi objek yang terdapat pada suatu gambar (Sultana *et al*. 2019). Dengan pendekatan metode pengolahan citra secara tradisional, *object detection* dapat dilakukan dengan berbagai teknik seperti *frame differencing*, *optical flow*, dan *background subtraction* (Panchal *et al*. 2015). Setiap teknik memiliki kelebihan dan kekurangan masing-masing dalam aspek akurasi maupun waktu komputasi (Prajapati dan Galiyawala 2015). Beberapa contoh studi kasus yang menerapkan teknik *object detection* tersebut di antaranya yaitu untuk deteksi wajah (Aydin dan Othman 2017), aplikasi untuk mendukung *smart city* (Hu L dan Ni Q 2018), deteksi kendaraan (Anandhalli dan Baligar 2018), dan deteksi pemain dan bola pada video siaran tenis (Archana dan Geetha 2015). Dewasa ini, dengan ketersediaan data dalam jumlah besar dan didukung oleh teknologi perangkat keras komputer yang semakin maju, penyelesaian permasalahan *object detection* mulai beralih pada pendekatan *deep learning* yang mana dalam berbagai penelitian terbukti menghasilkan akurasi yang lebih menjanjikan (Trigueros *et al*. 2018). Selain itu, *object detection* dengan *deep learning* dapat dilakukan secara otomatis oleh mesin karena tidak melewati tahapan ekstraksi fitur secara *hand-crafted*. Menurut Sultana *et al*. (2019) terdapat dua pendekatan dalam algoritme *object detection* berbasis *deep learning*, yaitu *two-stage* (dua tahap) dan *one-stage* (satu tahap) sebagaimana pada Gambar 2.1. Pada *two-stage object detector*, tahap *region proposal* dan klasifikasi dilakukan pada *network* yang terpisah. Pada pendekatan *one-stage object detection* ini, metode *region proposal* digunakan untuk mencari *region* atau bagian dari suatu gambar yang kemungkinan adalah sebuah objek. Kemudian dilakukan ekstraksi fitur dari setiap *region* tersebut menggunakan CNN. Hasil ekstraksi fitur tersebut dijadikan *input* untuk model *classifier* untuk proses klasifikasi sehingga diperoleh kelas dari *region* tersebut dan model *regressor* untuk memperoleh *bounding box* untuk objek yang terdapat pada gambar (Javier 2017). Beberapa algoritme *two-stage object detector* di antaranya adalah R-CNN (Girshick *et al*. 2014), Fast R-CNN (Girshick 2015), Faster R-CNN (Ren *et al*. 2017), dan R-FCN (Li *et al*. 2016). Algoritme seperti ini memiliki nilai presisi yang tinggi namun memiliki kekurangan yaitu kompleksitas komputasi yang tinggi (He *et al*. 2019).



Gambar 2.1 *Two-stage object detection*

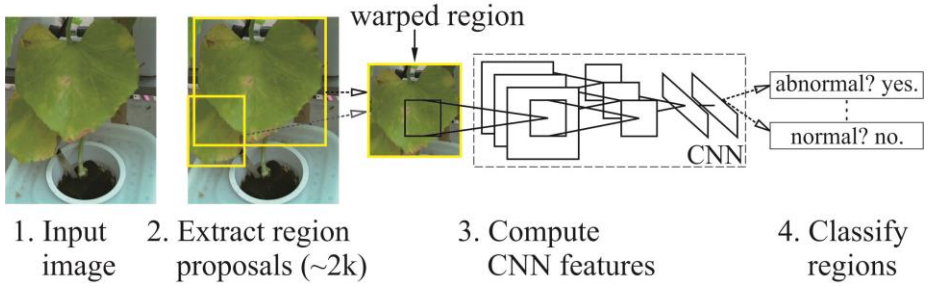
Dalam *one-stage object detector*, proses *region proposal* dan klasifikasi digabungkan dalam satu *network*, seperti YOLO (Redmon *et al*. 2016), YOLOv2 (Redmon dan Farhadi 2016), YOLOv3 (Redmon dan Farhadi 2018), dan SSD (Liu *et al*. 2016). Kelebihan algoritme *one-stage object detector* yaitu memiliki waktu

komputasi yang lebih kecil tetapi dengan tingkat akurasi yang lebih kecil daripada model *two-stage object detector* (He *et al*. 2019). Namun, beberapa *state-of-the-art* dari *one-stage detector* seperti YOLOv3 dan SSD telah dapat bersaing bahkan mengungguli metode *two-stage detector* dalam hal akurasi.

2.1.1 R-CNN

Sebelum R-CNN dipublikasikan, pada tahun 2014 OverFeat berhasil menjadi algoritme dengan kinerja terbaik pada dataset ILSVRC2013 untuk tugas *object detection* yang memiliki 200 kelas serta nilai mAP (*mean average precision*) yang dicapai adalah 24,3% (Sermanet *et al*. 2014). Algoritme OverFeat menggunakan pendekatan *sliding window* untuk mendeteksi objek pada suatu citra. Secara komputasional pendekatan *sliding window* sangat besar karena *window* menelusuri keseluruhan citra dengan berbagai ukuran dan rasio aspek sehingga menghasilkan begitu banyak citra-citra yang lebih kecil yang kemudian dimasukkan pada jaringan konvolusi.

R-CNN (*Region-based Convolutional Neural Network*) datang dengan mengganti pendekatan *sliding window* dengan *selective search* untuk membuat *region proposal* yang kemudian dimasukkan pada jaringan konvolusi. R-CNN merupakan pionir dalam metode *two-stage object detection*. Metode ini menghasilkan sekitar 2000 *region proposal* untuk setiap citra, mengekstrak vektor fitur dari setiap *proposal* menggunakan CNN, kemudian melakukan klasifikasi pada setiap *region* menggunakan metode *Support Vector Machine* (SVM) lalu metode regresi dilakukan untuk memprediksi *bounding box* baru untuk deteksi objek sebagaimana digambarkan pada Gambar 2.2 (Girshick *et al*. 2014). Percobaan pada dataset ILSVRC2013 menunjukkan bahwa R-CNN secara signifikan mengungguli OverFeat, dengan mAP 31,4% untuk R-CNN dibandingkan dengan mAP untuk OverFeat yaitu 24,3% (Girshick *et al*. 2014).



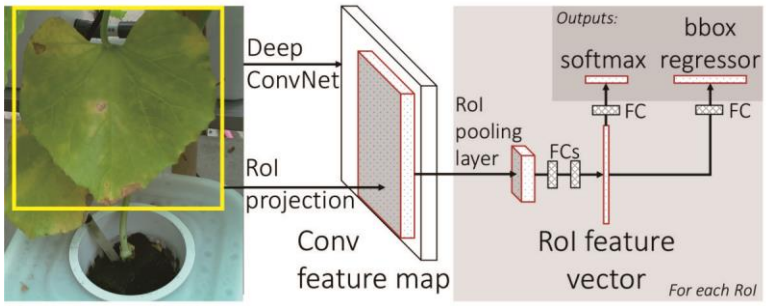
Gambar 2.2 Alur R-CNN

2.1.2 Fast R-CNN

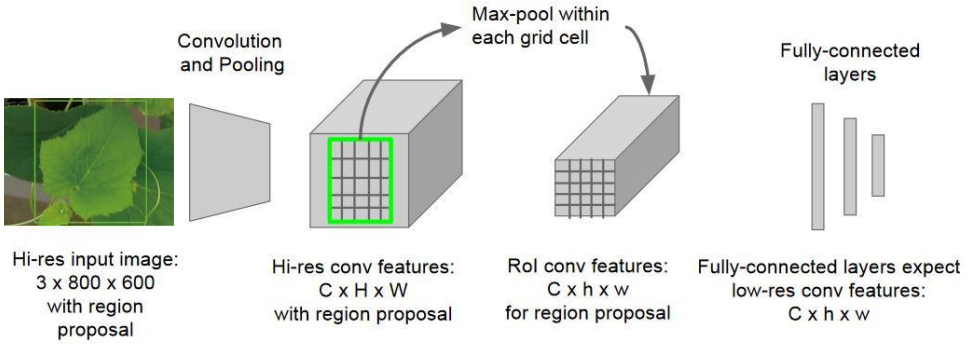
R-CNN memiliki beberapa kekurangan seperti memiliki alur pelatihan yang bertingkat (*multi-stage*) dan kompleks, yakni melalui tahap ekstraksi fitur untuk setiap *object proposal*, lalu melakukan klasifikasi menggunakan SVM dan regresi untuk menghasilkan *bounding box*; lambat pada saat pengujian (mendeteksi objek) karena harus melakukan ekstraksi fitur dengan menjalankan model CNN untuk setiap *region proposal*; dan membutuhkan sumber daya baik penyimpanan (*space*) dan waktu (*time*) yang besar, karena untuk melakukan prediksi kelas (menggunakan SVM) dan regresi terhadap

bounding box, fitur yang telah diekstrak dari setiap *region proposal* disimpan terlebih dahulu dalam penyimpanan (*disk*) (Girshick 2015).

Untuk memperbaiki kekurangan pada R-CNN, Girshick (2015) mengusulkan perubahan pada prosedur pelatihan yang *multi-stage* sehingga dapat dilatih secara bersamaan. Seperti pada Gambar 2.3, Fast R-CNN mengambil keseluruhan citra dan sejumlah *object proposal* sebagai *input*. *Input* diproses pada jaringan dengan beberapa *layer* konvolusi dan *max pooling* untuk menghasilkan *feature map*. Ukuran atau panjang dari *feature map* ini berbeda-beda untuk setiap *object proposal*, sedangkan sekuens *fully connected* (fc) *layer* memiliki ukuran yang sudah tentu. Oleh karena itu, *region of interest* (RoI) *pooling layer* diterapkan untuk mengekstrak vektor fitur dengan ukuran tertentu dari *feature map* tersebut sehingga diperoleh vektor fitur dengan ukuran yang sama dari setiap *feature map* (Gambar 2.4). Kemudian setiap vektor fitur (yang telah diekstrak melalui RoI *pooling layer*) dimasukkan pada fc *layer* yang pada akhirnya menghasilkan prediksi kelas objek (klasifikasi) menggunakan probabilitas *softmax* dan posisi *bounding box* (regresi) untuk setiap kelas objek (Girshick 2015).



Gambar 2.3 Alur Fast R-CNN



Gambar 2.4 RoI *pooling*

Hasil eksperimen menunjukkan bahwa Fast R-CNN menghasilkan tingkat presisi yang lebih tinggi serta waktu komputasi yang lebih cepat. Eksperimen dilakukan di antaranya menggunakan data *Visual Object Classes Challange* 2012 (VOC2012). Pada dataset VOC2012 tersebut Fast R-CNN memperoleh hasil terbaik dengan mAP 65,7% (dan 68,4% dengan data tambahan). Selain itu, Fast R-CNN bekerja dua kali lebih cepat daripada R-CNN. Demikian halnya eksperimen menggunakan dataset VOC2007 dan VOC2010, Fast R-CNN bekerja lebih baik daripada R-CNN (Girshick 2015).

Waktu komputasi yang dibutuhkan untuk pelatihan Fast R-CNN menggunakan jaringan VGG16 adalah 9,5 jam yang mana 8,8 kali lebih cepat daripada R-CNN yang membutuhkan waktu 84 jam untuk pelatihan menggunakan jaringan yang sama. Untuk waktu pengujian, Fast R-CNN membutuhkan 0,32 detik per gambar yang mana 146 kali lebih cepat dari R-CNN yang membutuhkan waktu 47 detik per gambar (Girshick 2015).

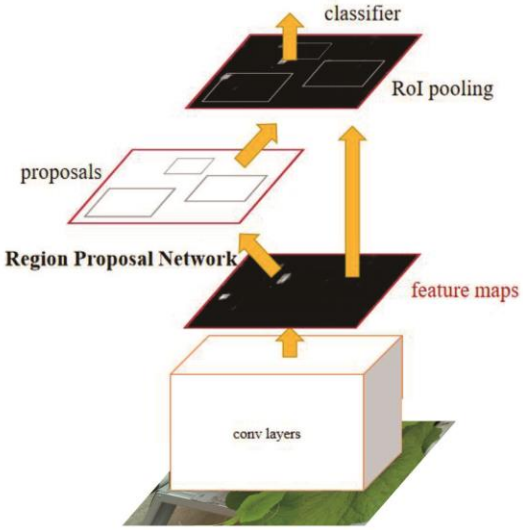
2.1.3 *Intersection over Union* (IOU)

Dalam *object detection*, *Intersection over Union* (IOU) digunakan untuk mengevaluasi *object detector* dengan mengukur kemiripan *predicted bounding box* dengan *ground truth* (GT) *bounding box*. Penghitungan metrik IOU dilakukan dengan membandingkan area irisan atau *overlapping region* dengan area gabungan antara *predicted bounding box* dengan *ground truth bounding box*. Adapun IOU antara bentuk (volume) $A. B \subseteq \mathbb{S} \in \mathbb{R}^n$ diperoleh dengan (Rezatofighi *et al.* 2019):

$$IOU = \frac{|A \cap B|}{|A \cup B|} \quad (1)$$

2.1.4 Faster R-CNN

Faster R-CNN merupakan perbaikan dari Fast R-CNN yang lebih efisien dalam aspek waktu komputasi dan memiliki kinerja deteksi hampir secara *realtime*. Perubahan mendasar pada Faster R-CNN adalah pada penggunaan *Region Proposal Network* (RPN) sebagai *region proposer* untuk membangkitkan *region proposal* menggantikan metode *selective search* (Gambar 2.5).



Gambar 2.5 Alur Faster R-CNN

RPN secara signifikan mereduksi waktu komputasi untuk membangkitkan *region proposal* karena adanya *computation sharing* dengan jaringan Fast R-CNN. Adapun untuk mendeteksi objek digunakan struktur jaringan Fast R-CNN (Ren *et al.* 2017). Dengan kata lain, Faster R-CNN adalah gabungan dari Fast R-CNN sebagai *object detector* dan RPN sebagai *region proposer*. Tabel 2.1 menunjukkan perbandingan kecepatan pengujian

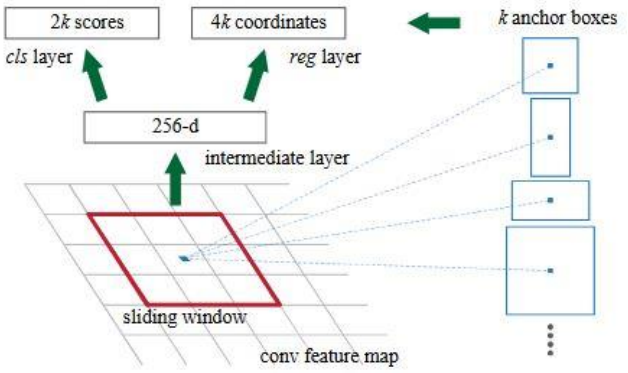
untuk R-CNN, Fast R-CNN, dan Faster R-CNN dengan R-CNN sebagai *benchmark* disertai nilai mAP masing-masing algoritme pada dataset VOC2007.

Tabel 2.1 Perbandingan kinerja R-CNN, Fast R-CNN, dan Faster R-CNN

	R-CNN	**Fast R-CNN**	**Faster R-CNN**
Waktu uji per gambar	50 detik	2 detik	0,2 detik
Speed up	1×	25×	250×
mAP (VOC2007)	66,0%	66,9%	66,9%

2.1.5 *Region Proposal Network* (RPN)

Region Proposal Network (RPN) merupakan gagasan utama dari Faster R-CNN. RPN bertujuan untuk mereduksi waktu komputasi yang sangat besar pada penggunaan *selective search* yang digunakan pada R-CNN dan Fast R-CNN untuk membangkitkan *region proposal*. *Input* untuk RPN adalah gambar (dengan berbagai ukuran) dan output yang dihasilkan adalah sejumlah *object proposal* yang masing-masing disertai dengan *objectness score* (skor yang menunjukkan keberadaan objek pada *object proposal*) dan *bounding box* (*x*, *y*, *w*, *h*). Proses RPN dapat dilihat pada Gambar 2.6.



Gambar 2.6 *Region Proposal Network* (RPN)

RPN menggunakan beberapa kotak (*box*) untuk memprediksi *region proposal* atau keberadaan objek (objek atau bukan objek) pada gambar. Kotak-kotak ini sejumlah banyaknya *proposal* yang mungkin untuk setiap lokasi pada *sliding-window*. Kotak-kotak ini disebut dengan istilah *anchor*. *Anchor* diletakkan di tengah *sliding-window*, dengan berbagai skala dan rasio aspek untuk menyesuaikan berbagai kemungkinan ukuran dari objek yang ada pada gambar (Ren *et al.* 2017). Misalkan untuk setiap lokasi digunakan *anchor* dengan 3 skala (128^2, 256^2, 512^2) dan 3 rasio aspek (1:1, 1:2, 2:1) sehingga terdapat 9 *anchor* untuk membangkitkan *region proposal*.

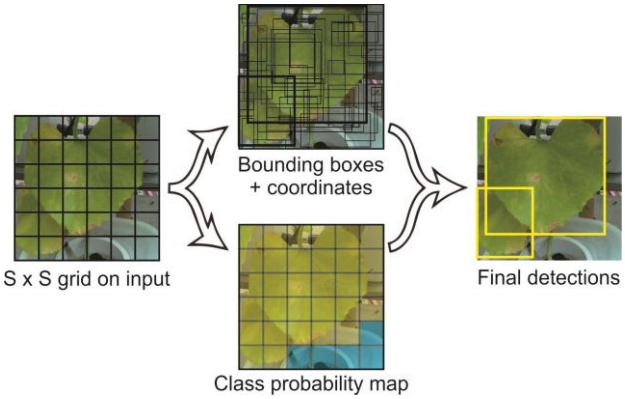
Terdapat dua kelas label *anchor* dalam pelatihan RPN, yaitu positif dan negatif. Kelas positif menunjukkan adanya latar depan (*background*) pada *anchor*, sedangkan kelas negatif menunjukkan adanya latar belakang (*background*) pada *anchor*. Suatu *anchor* termasuk kelas positif jika: 1) memiliki nilai IOU paling besar dan 2) memiliki nilai IOU lebih dari 0,7. Adapun *anchor* negatif yaitu *anchor* yang yang memiliki rasio IOU lebih kecil dari 0,3. *Anchor* yang tidak termasuk kelas positif dan negatif tidak diikutsertakan pada proses pelatihan model (Ren *et al.* 2017).

Dengan berbagai ukuran *anchor*, maka akan menghasilkan sejumlah banyak *region proposal* jika setiap *anchor* dipertimbangkan. Oleh karena itu, untuk mereduksi jumlah *region proposal*, dilakukan dua teknik yaitu: 1) mengabaikan *anchor* yang keluar dari batas gambar (*cross-boundary anchor*), dan 2) menerapkan *non-maximum supression* (NMS) pada *region proposal* dengan nilai ambang batas IOU 0,7 sehingga menghilangkan redundansi (Ren *et al.* 2017).

2.1.6 YOLO: *You Only Look Once*

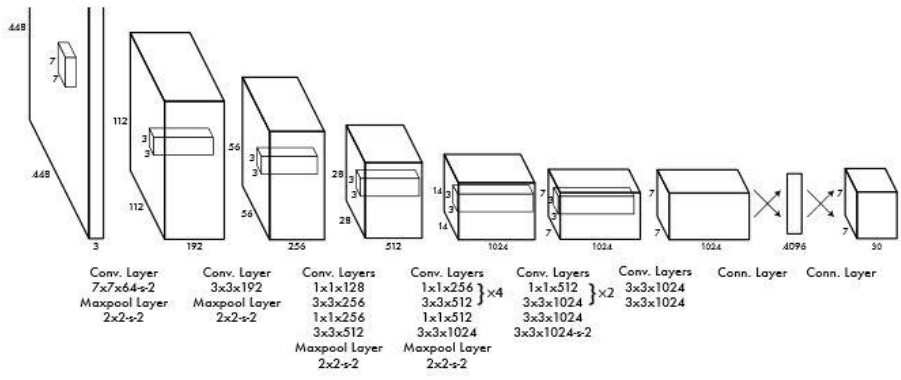
YOLO (*You Only Look Once*) merupakan salah satu algoritme *one-stage object detection*. YOLO memiliki kecepatan komputasi yang sangat tinggi dan dapat memproses gambar secara *real time*. Pada YOLO, komponen-komponen *object detection* yang terpisah disatukan dalam satu jaringan saraf tiruan. Sehingga YOLO memungkinkan pelatihan secara *end-to-end* dan memperoleh kecepatan yang tinggi dengan tetap mempertahankan presisi yang cukup tinggi (Redmon *et al.* 2016).

Secara umum model dan arsitektur jaringan YOLO dapat dilihat pada Gambar 2.7 dan Gambar 2.8. YOLO membagi gambar menjadi S x S *grid*.



Gambar 2.7 Model YOLO

Jika pusat suatu objek berada pada sel *grid* tertentu, maka sel *grid* tersebut bertanggung jawab untuk mendeteksi objek tersebut. Setiap sel *grid* memprediksi sejumlah *B bounding box*. YOLO akan memprediksi kelas dari objek yang terdeteksi beserta *bounding box*nya. Setiap *bounding box* memprediksi 5 variabel, yaitu *x*, *y*, *w*, *h*, dan nilai kepercayaan (*confidence*). Koordinat *x* dan *y* merepresentasikan titik pusat suatu objek, sedangkan *width*



Gambar 2.8 Arsitektur YOLO

(*w*) dan *height* (*h*) adalah lebar dan tinggi *bounding box* dari keseluruhan objek. Adapun nilai *confidence* merepresentasikan IOU antara *predicted box* dan *ground truth box* jika terdeteksi ada objek.

2.1.7 YOLOv2

YOLO menghasilkan *localization error* yang relatif tinggi dan *recall* yang relatif rendah dibandingkan dengan metode *object detection* berbasis *region proposal*. Selain itu, jumlah objek yang dapat dideteksi oleh YOLO terbatas karena setiap sel *grid* hanya dapat mendeteksi satu kelas objek. Sehingga YOLO tidak dapat mendeteksi lebih dari satu objek jika terdapat beberapa objek yang berdekatan dalam satu sel *grid*. Oleh karena itu, Redmon dan Farhadi (2017) mencoba mengatasi permasalahan tersebut dengan menerapkan beberapa perbaikan pada YOLO dan menamai metode yang diperbaiki ini dengan YOLOv2. Adapun beberapa perubahan pada YOLOv2 di antaranya adalah sebagai berikut (Redmon dan Farhadi 2017).

a) *Batch Normalization*

Batch normalization dilakukan dengan melakukan normalisasi terhadap nilai bobot *layer input* pada *artificial neural networks* (ANN) Teknik ini digunakan untuk meningkatkan kecepatan, kinerja, dan kestabilan ANN. Dengan menambahkan *batch normalization* pada YOLO, YOLOv2 memperoleh peningkatan mAP lebih dari 2%.

b) *Classifier* Resolusi Tinggi

YOLO melatih model *classifier* dengan resolusi 224 × 224 kemudian meningkatkan resolusinya menjadi 448 untuk deteksi. Adapun untuk YOLOv2, sebelum tahap pelatihan model untuk *object detection*, terlebih dahulu dilakukan *fine tune* pada model *classifier* dengan resolusi 448 × 448 sebanyak 10 epoch dengan dataset ImageNet. Dengan jaringan *classifier* resolusi tinggi ini, YOLOv2 memperoleh peningkatan mAP mendekati 4%.

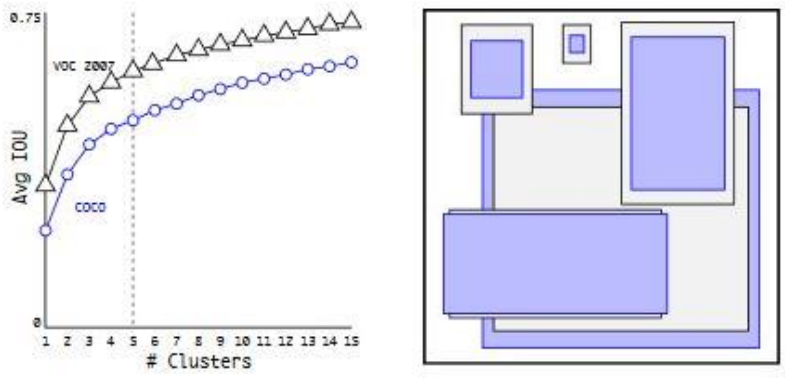
c) *Anchor/Prior Box* dan *Dimension Clusters*

Telah disebutkan sebelumnya bahwa pada YOLO jika titik tengah suatu objek jatuh pada suatu sel *grid* maka sel *grid* tersebut bertanggung jawab untuk memprediksi objek tersebut. Akan tetapi, setiap sel *grid* hanya dapat memprediksi satu objek. Hal ini menjadi kelemahan YOLO ketika terdapat lebih dari satu objek yang titik tengahnya jatuh pada sel *grid* yang sama. Oleh karena itu, untuk mengatasi masalah ini YOLOv2 mencoba untuk memungkinkan setiap sel *grid* memprediksi lebih dari satu objek dengan menggunakan *k bounding box*.

Untuk memprediksi *k bounding box* YOLOv2 menggunakan ide *anchor box* sebagaimana pada Faster R-CNN. Perbedaannya adalah jika pada Faster R-CNN *anchor* dipilih secara manual (baik jumlah, skala, dan rasionya), YOLOv2 mencoba untuk menemukan bentuk *anchor box* terbaik agar lebih memudahkan jaringan ketika proses pelatihan model deteksi. *Anchor* ini kemudian disebut juga dengan *prior* (Redmon dan Farhadi 2016). Metode yang digunakan untuk memilih *prior box* terbaik adalah *k-means clustering*. *K-means* diterapkan pada dataset *bounding box* data latih. IOU digunakan sebagai fungsi jarak menggantikan fungsi Euclidean yang biasa digunakan pada *k-means*. Sehingga, untuk matrik jarak digunakan yaitu sebagaimana pada Formula 2.

Percobaan dilakukan dengan menjalankan k-means dengan nilai *k* yang berbeda-beda kemudian dibuat plot rata-rata IOU dengan sentroid terdekat sebagaimana dapat dilihat pada Gambar 2.9. Pemilihan nilai *k* didasarkan pada *tradeoff* yang baik antara kompleksitas model dan *recall* yang tinggi (Redmon dan Farhadi 2017).

$$d(\text{box. centroid}) = 1 - IOU(\text{box. centroid}) \qquad (2)$$



(a) (b)

Gambar 2.9 *Clustering box dimensions* pada VOC dan COCO
(a) grafik jumlah *cluster* terhadap rata-rata IoU dan
(b) sentroid dengan 5 *cluster* untuk VOC dan COCO

d) Arsitektur Jaringan

YOLOv2 menggunakan arsitektur jaringan Darknet-19 (Tabel 2.2) sebagai *backbone*. Darknet-19 merupakan model klasifikasi yang dikembangkan untuk basis YOLOv2. Darknet-19 hanya membutuhkan 5,58 miliar operasi untuk memproses suatu gambar namun memperoleh hasil yang baik. Pada dataset ImageNet, Darknet-19 menghasilkan nilai akurasi *top-5* 91,2% yang mana lebih baik dari VGG-16 yang memperoleh 90% dan model *custom* YOLO yang memperoleh 88%.

2.1.8 YOLOv3

YOLOv3 merupakan generasi ketiga dari YOLO yang dipublikasikan pada 2018 oleh Redmon dan Farhadi. Terdapat beberapa perubahan pada YOLOv3 dari generasi sebelumnya (YOLOv2) di antaranya adalah sebagai berikut (Redmon dan Farhadi 2018).

a) Prediksi *Bounding Box*

Sebagaimana YOLOv2, YOLOv3 menggunakan *dimension clusters* sebagai *prior box*. Yang berubah pada YOLOv3 adalah perhitungan fungsi *cost*. Setiap *bounding box* memiliki *objectness score* yang diprediksi menggunakan *logistic regression*. Jika *bounding box prior* (*anchor*) tumpang tindih (*overlap*) dengan *ground truth object* lebih dari *bounding box prior* yang lainnya (memiliki nilai IOU paling besar), maka *objectness score*nya adalah 1. Untuk *prior* lainnya jika memiliki nilai IOU lebih besar dari ambang batas tertentu (misalnya 0,5) maka diabaikan.

Sistem hanya menetapkan satu *bounding box prior* untuk setiap *ground truth object*. Jika suatu *bounding box prior* tidak ditetapkan pada *ground truth object* manapun, maka tidak ada *loss* untuk prediksi koordinat dan kelas, hanya *objectness*.

Tabel 2.2 Darknet-19

Type	*Filters*	*Size/Stride*	*Output*
Convolutional	32	3 × 3	224 × 224
Maxpool		2 × 2 / 2	112 × 112
Convolutional	64	3 × 3	112 × 112
Maxpool		2 × 2 / 2	56 × 56
Convolutional	128	3 × 3	56 × 56
Convolutional	64	1 × 1	56 × 56
Convolutional	128	3 × 3	56 × 56
Maxpool		2 × 2 / 2	28 × 28
Convolutional	256	3 × 3	28 × 28
Convolutional	128	1 × 1	28 × 28
Convolutional	256	3 × 3	28 × 28
Maxpool		2 × 2 / 2	14 × 14
Convolutional	512	3 × 3	14 × 14
Convolutional	256	1 × 1	14 × 14
Convolutional	512	3 × 3	14 × 14
Convolutional	256	1 × 1	14 × 14
Convolutional	512	3 × 3	14 × 14
Maxpool		2 × 2 / 2	7 × 7
Convolutional	1024	3 × 3	7 × 7
Convolutional	512	1 × 1	7 × 7
Convolutional	1024	3 × 3	7 × 7
Convolutional	512	1 × 1	7 × 7
Convolutional	1024	3 × 3	7 × 7
Convolutional	1000	1 × 1	7 × 7
Averagepool		Global	1000
Softmax			

b) Prediksi Kelas

Generasi YOLO sebelumnya menggunakan fungsi *softmax* untuk klasifikasi sehingga setiap objek hanya ditetapkan pada satu kelas. Pendekatan ini berjalan jika diasumsikan bahwa *output* objek bersifat *mutually exclusive*. Akan tetapi, jika suatu objek memiliki lebih dari satu kelas (multi-label), maka diperlukan pendekatan klasifikasi multi-label untuk menghasilkan model yang lebih baik. Misalnya, suatu objek dapat memiliki dua label yaitu "*person*" dan "*woman*". Untuk melakukan klasifikasi multi-label, YOLOv3 mengganti fungsi *softmax* dengan *independent logistic classifier*. Adapun *loss function* untuk prediksi kelas yang digunakan selama pelatihan adalah *binary cross-entropy*.

c) Prediksi Multiskala

YOLO dan YOLOv2 memprediksi *output* pada *layer* terakhir, sedangkan YOLOv3 memprediksi *bounding box* pada beberapa skala yang berbeda.

Pada percobaan yang dilakukan oleh Redmon dan Farhadi (2018) menggunakan dataset COCO (*Common Objects in Context*), YOLOv3 memprediksi *bounding box* pada 3 skala yang berbeda. Setelah melakukan prediksi *bounding box* pada skala pertama, YOLOv3 mengambil *feature map* dari 2 *layer* terakhir dan melakukan *upsample* 2 kali. YOLOv3 juga mengambil *feature map* dari *layer* sebelumnya kemudian menggabungkannya dengan *feature map* yang sudah dilakukan *upsample*. Selanjutnya deteksi kedua dilakukan pada gabungan *feature map* tersebut dengan skala kedua. Untuk deteksi terakhir dengan skala ketiga, dilakukan proses yang sama dengan desain skala kedua.

Untuk menentukan *prior box*, Redmon dan Farhadi (2018) menggunakan k-means dan dipilih sebanyak 9 *cluster*. Karena pada pelatihan YOLOv3 digunakan 3 skala, maka masing-masing skala menggunakan 3 *prior box*. Pada dataset COCO, 9 *cluster* tersebut adalah: (10×13), (16×30), (33×23), (30×61), (62×45), (59×119), (116×90), (156×198), (373×326).

d) *Feature Extractor*

Untuk ekstraksi fitur YOLOv3 menggunakan pendekatan hybrid antara jaringan Darknet-19 dan jaringan residual (Tabel 2.3). Jaringan ini memiliki 53 *layer* dan dinamai Darknet-53. Jaringan ini lebih *powerful* daripada Darknet-19 tetapi masih lebih efisien dari ResNet-101 atau ResNet-152 (Redmon dan Farhadi 2018).

Tabel 2.3 Darknet-53

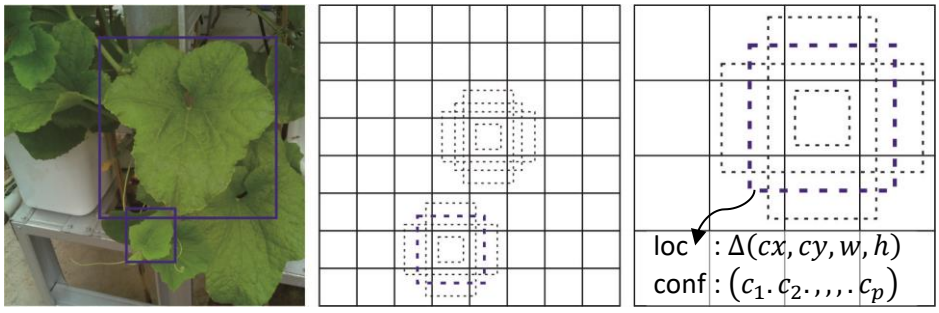
	Type	*Filters*	*Size/Stride*	*Output*
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Averagepool		Global	
	Connected		1000	
	Softmax			

2.1.9 SSD: *Single Shot Multibox Detector*

SSD (*Single Shot Multibox Detector*) merupakan metode *one-stage object detection* untuk mendeteksi objek pada gambar dengan menggunakan *single deep neural network*. *Object detector* seperti Faster R-CNN memiliki akurasi yang cukup tinggi tetapi membutuhkan waktu komputasi yang lama sehingga belum dapat digunakan secara *real time*. Sebaliknya, YOLO yang merupakan *state-of-the art* untuk *single shot detector* memiliki kinerja secara *real time* tetapi mengorbankan tingkat akurasi. Untuk memperbaiki *tradoff* tersebut, Liu *et al*. (2016) mengembangkan metode SSD yang lebih cepat namun memiliki tingkat akurasi yang secara signifikan lebih tinggi dari YOLO bahkan menyamai akurasi metode *two-stage detection* termasuk Faster R-CNN.

Sebagaimana *object detector* lainnya, SSD memprediksi *bounding box* disertai label kelas untuk objek yang terdeteksi pada suatu gambar. Struktur jaringan SSD diawali dengan lapisan-lapisan (*layers*) ekstraksi fitur dari arsitektur jaringan yang biasa digunakan untuk klasifikasi citra (seperti VGG, Inception, ResNet, dan sebagainya), yakni tidak termasuk bagian *layers* untuk klasifikasi. Jaringan ini dalam SSD disebut sebagai *base network*. Di akhir *base network* kemudian ditambahkan beberapa *convolutional feature layers* yang ukurannya semakin kecil dari *layer* ke *layer* sehingga memungkinkan untuk melakukan prediksi deteksi pada multiskala. Pada setiap *feature layer* (tambahan) tersebut dilakukan prediksi deteksi dengan model konvolusi yang berbeda (Liu *et al.* 2016). Gambar 2.13 menunjukkan struktur jaringan SSD dengan ukuran *input* 300 × 300.

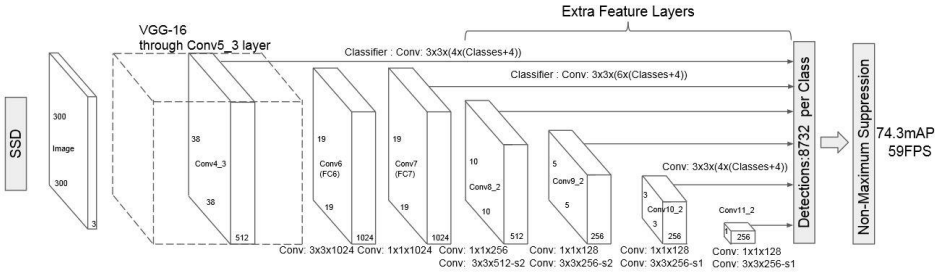
Untuk memprediksi *bounding box* SSD menggunakan *prior* yang disebut sebagai *default box*. *Default box* ini mirip seperti *anchor* pada Faster R-CNN, bedanya adalah pada SSD *default box* ini diterapkan pada beberapa *feature map* dengan resolusi yang berbeda. Satu set *default box* diterapkan pada setiap sel *feature map*. Di setiap sel *feature map*, SSD memprediksi *offsets* (*x*, *y*, *w*, *h*) berdasarkan bentuk *default box* di setiap sel, dan skor per kelas yang menunjukkan keberadaan objek kelas pada *box* tersebut. Proses ini membantu SSD meningkatkan akurasi deteksi pada objek berukuran kecil karena dapat meliputi bentuk dan ukuran objek *input* yang bermacam-macam. Misalnya, pada Gambar 2.12 objek daun yang lebih besar cocok dengan *default box* di *feature map* 4 × 4, tetapi tidak dengan satupun *default box* di *feature map* 8 × 8. Hal ini dikarenakan *default box* pada *feature map* 8 × 8



(a) (b) (c)

Gambar 2.10 *Framework* SSD. (a) Gambar dengan GT *box*, (b) 8 × 8 *feature map*, dan (c) 4 × 4 *feature map*

memiliki skala yang berbeda dan tidak terdapat kecocokan dengan objek daun yang lebih besar pada gambar, sehingga pada saat pelatihan dianggap sebagai negatif (tidak ada objek) (Liu *et al.* 2016).



Gambar 2.11 Struktur jaringan SSD

Default box pada SSD ditentukan untuk masing-masing *feature map*. Misalkan untuk melakukan prediksi digunakan sejumlah *m feature map*, maka *default box* diperoleh melalui perhitungan sebagai berikut.

1. Skala

Skala untuk *default box* pada *feature map* ke-*k* dihitung dengan Formula 3 (Liu *et al.* 2016).

$$s_k = s_{min} + \frac{s_{max} - s_{min}}{m - 1}(k - 1); \;\; k \in [1.m] \qquad (3)$$

di mana s_{min} adalah 0,2 dan s_{max} adalah 0,9, yang berarti *layer* deteksi terbawah memiliki skala 0.2 dan skala ini meningkat secara linear sesuai Formula 3 pada *layers* deteksi selanjutnya sampai skala 0,9 untuk *layer* deteksi terakhir.

2. Rasio aspek

SSD menggunakan lima rasio aspek yang berbeda untuk *default box* yang didefinisikan sebagai $a_r \in \left\{1.2.3.\frac{1}{2}.\frac{1}{3}\right\}$. Dengan demikian, lebar (*w*) dan tinggi (*h*) masing-masing *default box* dapat dihitung menggunakan Formula 4 dan 5 sebagai berikut (Liu *et al.* 2016).

$$w_k^a = s_k\sqrt{a_r} \qquad (4)$$

$$h_k^a = \frac{s_k}{\sqrt{a_r}} \qquad (5)$$

Kemudian untuk rasio aspek 1, SSD menambahkan satu *default box* dengan skala yang dihitung dengan Formula 6 sebagai berikut (Liu *et al.* 2016).

$$s'_k = \sqrt{s_k s_{k+1}} \qquad (6)$$

Sehingga secara total terdapat 6 *default box* yang digunakan untuk memprediksi objek pada setiap *feature map*.

3. Titik tengah (*center*)

Titik tengah masing-masing *default box* dihitung dengan Formula 7 (Liu *et al.* 2016).

$$x.y = \frac{i+0{,}5}{|f_k|}.\frac{j+0{,}5}{|f_k|};\ \ i.j \in [0.|f_k|) \qquad (7)$$

di mana $|f_k|$ adalah ukuran *feature map* ke-*k*.

Tabel 2.4 menampilkan beberapa penelitian tentang penerapan *object detection* berbasis *deep learning* terutama di bidang pertanian atau area objek yang padat. Semua pengukuran *inference time* pada Tabel 2.4 dilakukan di GPU (*graphics processing unit*).

Tabel 2.4 Beberapa *paper* tentang penelitian terkait

Sumber	Algoritme	Dataset	Hasil
Arsenovic *et al.* (2019)	Faster R-CNN	PlantDisease (penyakit tanaman)	mAP: 84,4% mAP: 91,8% (dengan metode *state-of-the-art*)
	YOLOv3		mAP: 0,812% mAP: 91,9% (dengan metode *state-of-the-art*)
	SSD513		mAP: 79,8% mAP: 92,5% (dengan metode *state-of-the-art*)
Lei *et al.* (2019)	SSD	DOTA (*Dataset for Object Detection in Aerial Images*)	mAP: 10,59%
	Faster R-CNN		mAP: 36,29%
	Modifikasi Faster R-CNN		mAP: 52,93% *inference time*: 0,24 s
	Modifikasi YOLOv3		mAP: 43,66% *inference time*: 0,088 s
Liu dan Wang (2020)	Faster R-CNN	Hama dan penyakit tomat	mAP: 90,67% *inference time*: 2,869 s
	SSD		mAP: 84,32% *inference time*: 0,026 s
	YOLOv3		mAP: 88,31% *inference time*: 0,021
	Modifikasi YOLOv3		mAP: 92,39% *inference time*: 0,020 s
Xie *et al.* (2020)	Faster R-CNN	GLDD (*Grape Leaf Disease Dataset*)	mAP: mencapai 71,5% *inference time*: dari 0,054 s sampai 0,141 s
Buzzy *et al.* (2020)	Faster R-CNN	Tanaman *Arabidopsis*	AP: 0,60% *inference time*: 0,918 s

2.2 Raspberry Pi

Perangkat dengan kemampuan komputasi yang terbatas memiliki sumber daya yang terbatas, biasanya terdiri dari prosesor *single-core* dan jumlah RAM

yang terbatas. Salah satu perangkat dengan kemampuan komputasi terbatas adalah Raspberry Pi. Raspberry Pi yaitu *single-board computer* berukuran kecil yang dapat digunakan untuk menjalankan berbagai program komputer dengan dihubungakan pada monitor dan menggunakan *keyboard* dan *mouse* standar.

Sebagaimana dilansir pada situs raspberrypi.org, Raspberry Pi dapat digunakan untuk menjelajah internet, memutar video berkualitas tinggi, hingga membuat *spreadsheet*, pemrosesan kata, dan bermain *game*. Meskipun memiliki kekurangan dalam kemampuan komputasi dibandingkan dengan sistem desktop, Raspberry Pi memiliki harga yang murah dan ukuran yang kecil sehingga lebih disukai untuk penggunaan pada kasus-kasus tertentu seperti sistem pengawasan dan *Internet of Things* (IoT). Selain itu, Raspberry Pi dapat melakukan tugas *computer vision* yang dapat bermanfaat dalam mendukung berbagai bidang seperti *smart cities*, pertanian, *smart home*, dan lain sebagainya (Yadav dan Kumari 2018).

Raspberry Pi memiliki berbagai model dengan spesifikasi yang berbeda-beda. Sampai Juni 2019, terdapat 4 keluarga Raspberry Pi mulai dari Raspberry Pi 1 sampai Raspberry Pi 4 dengan model-model dan spesifikasi yang beragam. Sebagaimana dilansir pada situs socialcompare.com, berbagai model Raspberry Pi memiliki kapasitas RAM paling kecil 256 MB dan paling besar 4 GB. Raspberry Pi 4 B adalah produk keluaran terbaru dari Raspberry dengan prosesor *quad-core* 64-bit ARM Cortex-A72 berkecepatan 1,5 GHz yang memiliki beberapa pilihan kapasitas RAM meliputi 1 GB, 2 GB, dan 4 GB. Adapun harga untuk pembelian Raspberry Pi berkisar antara $5 sampai $35.

2.3 Kelainan Daun Tanaman Melon

Penyakit tanaman sangat berpengaruh terhadap kerugian produksi dan ekonomi dalam sektor pertanian global (Reverchon *et al*. 2016). Salah satu tanaman yang sangat sensitif terhadap penyakit adalah *C. melo*. Tumbuhan *cucurbit* seperti *C. melo* merupakan spesies tanaman yang memiliki siklus pendek (80 – 110 hari) dan tumbuh dengan cepat sehingga bersifat lunak yang mana membuatnya sangat sensitif terhadap hama dan penyakit (Balliu dan Sallaku 2017). Dalam Keinath *et al*. (2017) disebutkan bahwa beberapa penyakit tanaman melon yang gejalanya dapat terlihat pada daun di antaranya adalah *powdery mildew*, *cucurbit yellow stunting disorder* (CSYD), dan *cucumber green mottle mosaic* (CGMM)..

Selain penyakit atau hama, kelainan dapat juga disebabkan oleh kekurangan asupan nutrisi pada tanaman. Kurangnya konsentrasi elemen nutrisi tertentu pada organ dan jaringan tanaman menimbulkan gejala yang dapat dicirikan secara fisik berdasarkan kekurangan nutrisi tersebut (Nerson 2008). Misalnya, sebagaimana dilansir pada situs advancednutrients.com, kekurangan nitrogen pada tanaman dapat dicirikan dengan daun berwarna hijau pucat, bahkan sampai kuning jika konsentrasi nitrogen sangat kurang; kekurangan fosfor ditandai dengan dau berwarna gelap, akar kecil, bunga sangat kecil, dan daun berwarna merah atau ungu; kekurangan magnesium ditandai dengan tepi daun yang menguning, tepi daun semakin kuning seiring semakin kurangnya konsentrasi magnesium, dan semakin banyak daun yang terpengaruh.

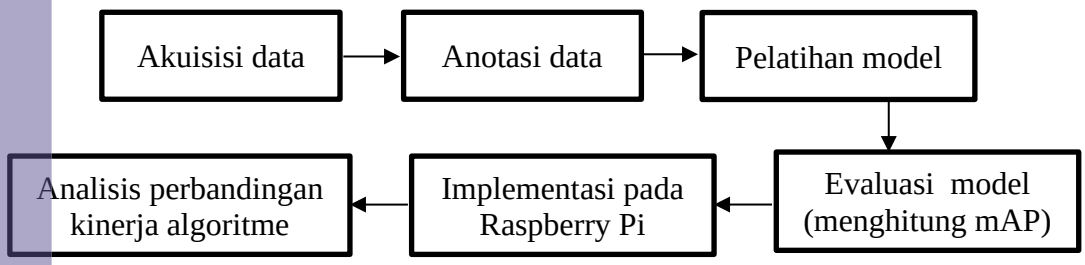
III METODE

3.1 Data dan Area Studi

Dalam penelitian ini, dataset yang digunakan meliputi data citra sekumpulan daun tanaman melon. Proses pengambilan citra dilakukan dengan memotret daun dan kumpulan daun tanaman melon menggunakan kamera Raspberry Pi. Dataset yang diperoleh digunakan untuk proses *object detection* dengan algoritme *deep learning* Faster R-CNN, YOLOv3, dan SSD. Ketiga algoritme tersebut dipilih karena selain disusun dengan baik, penggunaannya sangat luas baik di bidang akademik seperti pada penelitian-penelitian maupun secara praktis (Alganci *et al.* 2020; Li *et al.* 2020; Nguyen *et al.* 2020). Setelah model *deep learning* diperoleh melalui proses pelatihan, kemudian dilakukan evaluasi model dan analisis kinerja masing-masing algoritme tersebut pada Raspberry Pi.

3.2 Tahapan Penelitian

Terdapat beberapa tahapan yang dilakukan dalam penelitian ini sebagaimana pada Gambar 3.1. Secara umum, tahapan penelitian terdiri dari akuisisi data, anotasi data, pelatihan model *object detection*, evaluasi model (menghitung mAP), implementasi model pada Raspberry Pi, dan analisis perbandingan kinerja dari algoritme-algoritme *object detection* yang digunakan.



Gambar 3.1 Tahapan penelitian

3.3 Akuisisi Data

Data yang digunakan dalam penelitian ini adalah citra sekumpulan daun melon yang diambil dari iSurf Lab (*IoT for Smart Urban Farming Laboratory*) departemen Ilmu Komputer FMIPA IPB dan *greenhouse Agribusiness and Technology Park* (ATP) IPB Cikarawang. Pengambilan data dilakukan antara pagi menjelang siang sampai tengah hari (dimulai pukul 11.00 WIB sampai dengan selesai) ketika kondisi lingkungan memiliki pencahayaan yang cukup sehingga memungkinkan untuk melihat daun melon secara jelas. Usia tanaman pada saat pengambilan data yaitu sekitar masa pertumbuhan sampai dewasa. Gambar diambil menggunakan Raspberry Pi 4 (Lampiran 2) dan kamera Raspberry Pi Rev 1.3 (Lampiran 3). *Setup* alat akuisisi data gambar dapat dilihat pada Lampiran 4 dan 5.

3.4 Anotasi Data

Anotasi data dilakukan dengan memberikan *bounding box* dan label kelas pada setiap objek pada gambar. Dalam penelitian ini, data diklasifikasikan ke dalam dua kelas, yaitu memiliki kelainan (abnormal) dan tidak memiliki kelainan

(normal). Proses pemberian *ground truth box* dan label pada citra dilakukan menggunakan *tool* labelImg, suatu perangkat lunak yang berfungsi membuat *bounding box* dan label kelas pada citra untuk anotasi citra.

3.5 Pelatihan Model

Algoritme *object detection* dengan pendekatan *deep learning* dibagi menjadi dua jenis, yaitu *two-stage object detector* dan *one-stage object detector*. Pada *two-stage object detector* proses *region proposal* dan klasifikasi dilakukan pada *network* yang terpisah, sedangkan dalam *one-stage object detector* proses *region proposal* dan klasifikasi digabungkan dalam satu *network*. *Object detector* yang digunakan dalam penelitian ini meliputi *two-stage detector* yaitu Faster R-CNN dan *one-stage detector* yaitu SSD dan YOLOv3. Pelatihan (*training*) model dilakukan menggunakan *graphics processing unit* (GPU) dengan skema sebagai berikut.

a. Dataset
   Rasio data latih dan uji pada eksperimen ini adalah 0.8:0.2.
b. *Anchor*/*prior*/*default box*
   Untuk Faster R-CNN digunakan *anchor box* dengan tiga skala dan tiga rasio, sedangkan *prior box* untuk YOLOv3 diperoleh dari hasil *k-means clustering* pada data *bounding box* pada dataset citra. Adapun *default box* pada SSD ditentukan sesuai dengan perhitungan menggunakan Formula 3, 4, 5, 6, dan 7.
c. Augmentasi data
   Dengan melakukan proses augmentasi pada data dapat menambah jumlah data untuk pelatihan model sehingga dapat mengatasi permasalahan *overfitting* serta membuat model menjadi lebih *robust* dan memiliki akurasi yang lebih baik (Liu *et al.* 2016). Terdapat berbagai macam teknik augmentasi yang sering digunakan dalam *deep learning* untuk pengolahan citra di antaranya *random cropping*, *image mirroring*, dan *multi-scale training* (Zoph *et al.* 2019). Adapun teknik augmentasi yang digunakan pada penelitian ini adalah *random horizontal flipping*.
d. *Feature extractor*
   Faster R-CNN dan SSD menggunakan tiga *pretrained models* yaitu dengan *backbone* InceptionV2, ResNet50, dan MobileNet. Ketiga model tersebut sudah dilatih dengan dataset COCO. Adapun untuk YOLOv3 digunakan model *pretrained* dengan *backbone* Darknet-53 yang telah dilatih pada dataset COCO.
e. *Framework*
   Penelitian ini menggunakan Tensorflow sebagai *machine learning framework* untuk pelatihan Faster R-CNN, SSD dan YOLOv3.

3.6 Evaluasi Model

Evaluasi terhadap model deteksi dilakukan dengan menghitung *precision* dan *recall*. *Precision* dan *recall* digunakan untuk mengukur seberapa baik objek yang terdeteksi sesuai dengan objek referensi. *Recall* didefinisikan sebagai proporsi objek yang terdeteksi dengan benar di antara semua objek yang seharusnya terdeteksi. *Precision* adalah proporsi antara kelas positif yang terdeteksi dengan

benar di antara jumlah kelas positif yang terdeteksi (Godil *et al*. 2014). *Precision* dan *recall* dihitung dengan Formula 8 dan 9 berdasarkan Tabel 3.1 (He *et al*. 2019).

Tabel 3.1 *Confusion matrix*

	Positif sebenarnya	Negatif sebenarnya
Positif prediksi	*True Positive* (TP)	*False Positive* (FP)
Negatif prediksi	*False Negative* (FN)	*True Negative* (TN)

$$precision = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{8}$$

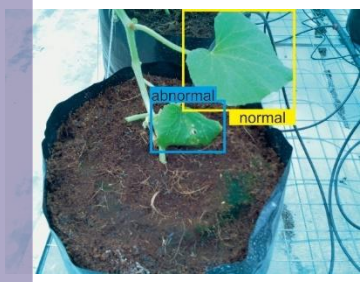
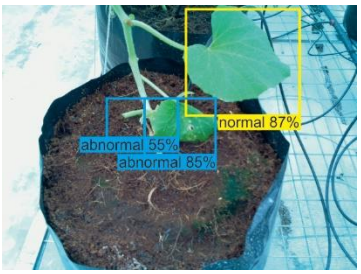
$$recall = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{9}$$

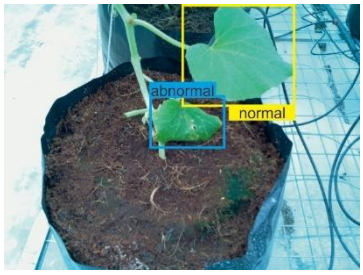
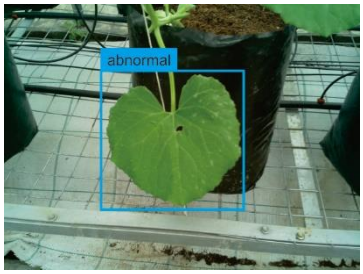
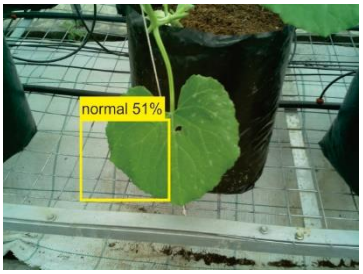
Nilai ambang batas IOU (IOU *threshold*) dapat digunakan untuk menentukan apakah suatu deteksi benar. Misalkan ditetapkan nilai ambang batas IOU 0,5, maka:

- Jika IOU ≥ 0,5, berarti deteksi benar dan termasuk *True Positive* (TP). Dengan kata lain, model memprediksi objek sebagai objek dengan benar.
- Jika IOU < 0,5, berarti deteksi salah dan termasuk *False Positive* (FP), atau model memprediksi *background* sebagai objek padahal seharusnya bukan objek.
- Ketika terdapat objek yang gagal dideteksi oleh model, maka termasuk *False Negative* (FN), artinya model mengira objek sebagai *background*.
- Ketika model tidak melakukan deteksi pada *background* yang memang tidak perlu dideteksi, maka termasuk *True Negative* (TN). Untuk *object detection*, metrik FN ini tidak digunakan.

Metrik untuk prediksi dihitung untuk masing-masing kelas. Tabel 3.2 menampilkan contoh perhitungan TP, FP, dan FN untuk kelas daun abnormal dengan nilai ambang batas IOU 0,5.

Tabel 3.2 Contoh perhitungan TP, FP, dan FN

Ground truth	Prediksi	Keterangan
		**Abnormal**: **1 TP** – daun abnormal terdeteksi dengan IOU > 0,5 **1 FP** – daun abnormal terdeteksi dengan IOU > 0,5, tetapi bukan skor IOU yang tertinggi **Normal**: **1 TP** – daun normal terdeteksi dengan IOU > 0,5

Ground truth	Prediksi	Keterangan
		**Abnormal**: **1 FN** – tidak ada *box* yang mendeteksi daun abnormal **Normal**: **1 FN** – tidak ada *box* yang mendeteksi daun normal
		**Abnormal**: **1 FN** – tidak ada *box* yang mendeteksi daun abnormal **Normal**: **1 FP** – tidak ada daun normal, tetapi model mendeteksi satu daun normal

Average Precision (AP) digunakan sebagai indikator untuk mengukur kinerja pada *object detection* dan untuk mengevaluasi kinerja dari kelas dataset pada model. AP meringkas bentuk kurva *precision*/*recall* dan mendefinisikan skornya berdasarkan rata-rata *precision* dari suatu set nilai *recall* yang berjarak sama (0, 0.1, 0.2, ..., 1). Penghitungan AP mengikuti Formula 10 dan 11 (He *et al*. 2019).

$$AP = \frac{1}{11} \sum_{r \in \{0.0, 1.0, 2., ,, 1\}} P_{interp}(r) \tag{10}$$

$P_{interp}(r)$ atau *interpolated precision* didefinisikan sebagai

$$P_{interp} = \max_{\tilde{r}:\tilde{r} \geq r} P(\tilde{r}) \tag{11}$$

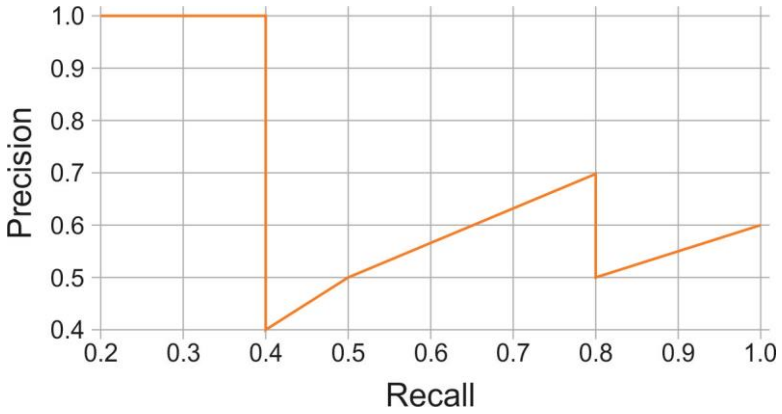
di mana $P(\tilde{r})$ adalah *precision* yang diukur pada *recall* ($\tilde{r}$).

Sebagai ilustrasi, misalkan terdapat dataset citra sekumpulan daun. Setelah melalui proses *object detection*, diperoleh hasil prediksi untuk kelas daun abnormal. Dari hasil prediksi, diperoleh nilai *precision* dan *recall* sebagaimana pada Tabel 3.2. Berdasarkan Tabel 3.2, nilai *recall* dan *precision* dapat disajikan dalam bentuk plot sebagaimana pada Gambar 3.2. Mengikuti Formula 11, nilai *interpolated precision* ditentukan berdasarkan nilai *precision* maksimum pada *recall* yang lebih besar (jika dilihat pada Gambar 3.3 maka menuju arah kanan). *Interpolated*

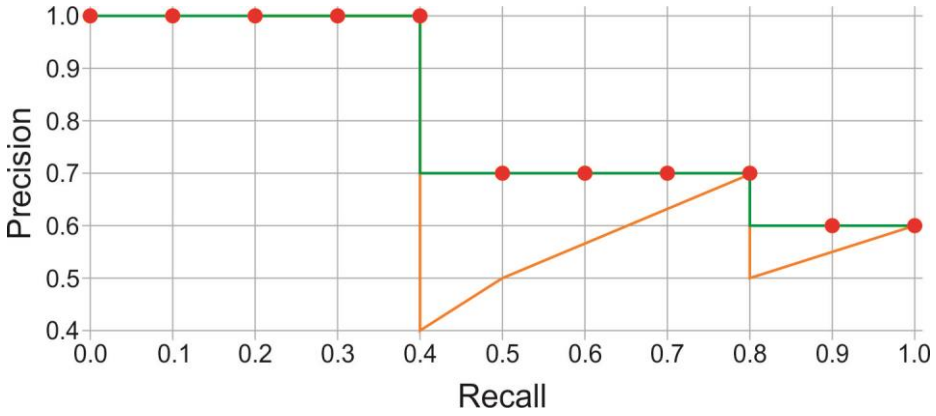
precision ini dihitung terhadap 11 nilai *recall* yang berjarak sama dari 0 sampai 1 (0, 0.1, 0.2, ..., 1.0) sebagaimana ditampilkan pada Gambar 3.3.

Tabel 3.3 Contoh nilai *recall* dan *precision*

Recall	*Precision*
0,2	1,0
0,4	1,0
0,4	0,7
0,4	0,5
0,4	0,4
0,5	0,5
0,8	0,7
0,8	0,5
1,0	0,6



Gambar 3.2 Plot *precision* terhadap *recall*



Gambar 3.3 *Interpolated precision*

Menggunakan Formula 10, maka pada contoh ini skor AP dapat dihitung:

$$AP = \frac{1}{11} \times \left(P_{interp}(0) + P_{interp}(0{,}1) + P_{interp}(0{,}2) + ,,, + P_{interp}(1{,}0) \right)$$

$$AP = \frac{1}{11} \times (1{,}0 + 1{,}0 + 1{,}0 + 1{,}0 + 1{,}0 + 0{,}7 + 0{,}7 + 0{,}7 + 0{,}7 + 0{,}6 + 0{,}6)$$

$$AP = \frac{1}{11} \times 9 = 0{,}82$$

Untuk evaluasi pada deteksi multi-kelas, rata-rata dari seluruh nilai AP disebut sebagai *mean average precision* (mAP) (He *et al*. 2019).

3.7 Implementasi pada Raspberry Pi

Model-model *object detector* yang telah dilatih kemudian diimplementasikan pada perangkat komputasi terbatas, yang dalam penelitian ini adalah Raspberry Pi, untuk mendeteksi kelainan daun tanaman melon. Pada Raspberry Pi dilakukan pengujian masing-masing model *object detector* menggunakan beberapa data uji. Pada tahap implementasi ini, dilakukan pengukuran terhadap *inference time* dan penggunaan sumber daya (seperti CPU dan memori) masing-masing algoritme pada Raspberry Pi. Dari proses *object detection* yang dilakukan pada situs resmi Tensorflow, yaitu tensorflow.org, dapat dipahami bahwa *inferenc time* merupakan waktu yang dibutuhkan model ketika menentukan atau mengambil kesimpulan tentang *bounding box* dan label kelas pada setiap objek yang terdeteksi pada suatu gambar. *Inference time* diukur pada saat model melakukan deteksi pada suatu gambar atau pada saat pengujian.

3.8 Analisis Perbandingan Kinerja Algoritme

Setiap algoritme *object detection* dibandingkan berdasarkan hasil evaluasi yang meliputi penghitungan mAP serta pengukuran waktu komputasi dan penggunaan sumber daya. Analisis dilakukan terhadap kinerja setiap algoritme sehingga dapat dijelaskan kelebihan dan kekurangan masing-masing algoritme untuk mendeteksi kelainan daun tanaman melon menggunakan Raspberry Pi berdasarkan *trade off* akurasi, waktu komputasi, dan penggunaan sumber daya.

3.9 Peralatan Penelitian

Peralatan yang digunakan dalam penelitian ini meliputi:

a. Perangkat keras
   1) Pelatihan model dilakukan menggunakan *High Performance Computing* (HPC) Lembaga Ilmu Pengetahuan Indonesia (LIPI) dengan rincian:
      - Jumlah Core CPU : 28
      - Jumlah GPU : 1
      - Kapasitas Penyimpanan : 100 GB

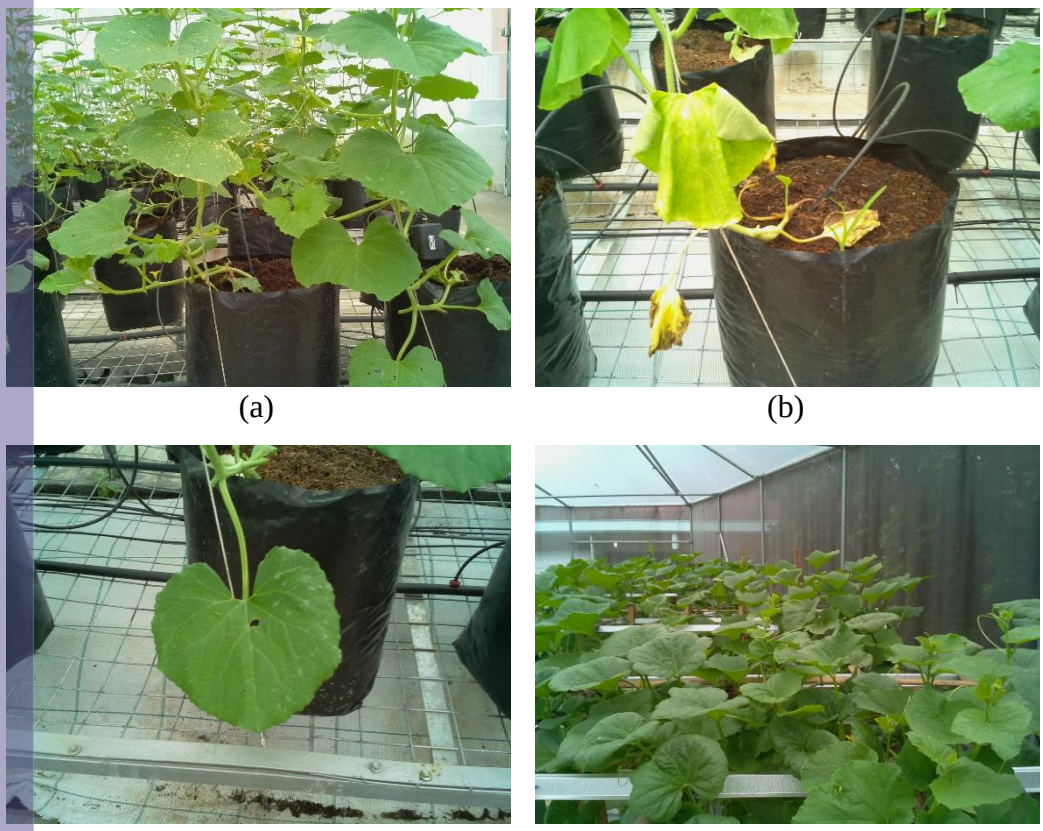
      Beberapa hal terkait dengan pengajuan penggunaan fasilitas HPC LIPI untuk penelitian ini dapat dilihat pada Lampiran 6.
   2) Raspberry Pi 4 dengan RAM 4 GB dan kapasitas penyimpanan 16 GB (Lampiran 2)
   3) Kamera Pi Rev 1.3 dengan resolusi gambar 5 MP (Lampiran 3)

b. Perangkat lunak
   1) Sistem Operasi Linux untuk komputer
   2) Sistem Operasi Raspbian untuk Raspberry Pi
   3) Python 3 sebagai bahasa pemrograman
   4) Tensorflow 1.0 sebagai *framework*

IV HASIL DAN PEMBAHASAN

4.1 Akuisisi Data

Proses akuisisi data di iSurf Lab dan ATP IPB dilakukan 3 kali di hari yang berbeda (pada 17 dan 20 Februari 2020 di iSurf Lab dan pada 19 Februari 2020 di ATP IPB) pada waktu yang sama (yaitu sekitar pukul 11.00 sampai 12.30 WIB). Gambar ditangkap menggunakan kamera Raspberry Pi (Lampiran 3) melalui sudut tangkapan yang berbeda-beda sehingga menghasilkan posisi daun yang beragam agar dapat melatih model *object detector* lebih baik. Beberapa gambar diambil dari arah depan tanaman dengan jarak 70 cm dari tempat tanam, sedangkan beberapa gambar lainnya diambil secara lebih dekat agar lebih jelas. Adapun beberapa gambar lainnya diambil dari arah atas dan samping daun. Dari proses pengambilan gambar tersebut diperoleh 522 data citra untuk dianotasi. Masing-masing data citra berukuran lebar dan tinggi berturut-turut adalah 2592 dan 1944 piksel.



(a) (b)

(c) (d)

Gambar 4.1 Contoh tangkapan kamera (a) dari depan dengan jarak 70 cm dari tempat tanam, (b) jarak lebih dekat, (c) dari atas, dan (d) dari samping.

4.2 Anotasi Data

Untuk setiap data citra dilakukan anotasi, yaitu pemberian *bounding box* dan label kelas pada objek pada gambar. Gambar 4.1 menampilkan contoh citra daun yang dikategorikan pada kelas abnormal (a) dan normal (b). Proses anotasi data citra menggunakan *tool* labelImg dapat dilihat pada Gambar 4.2.

Daun abnormal di antaranya ditandai dengan perubahan warna bagian atau keseluruhan daun seperti menjadi kekuningan, kecoklatan, atau hijau gelap, bagian tepi daun menjadi kekuningan, terdapat bagian daun yang berlubang atau sobek, atau terdapat bercak kecuali bercak putih bekas penyemprotan cairan seperti pestisida. Adapun daun dikategorikan sebagai normal jika tidak terdapat indikasi abnormal. Penentuan indikator abnormal pada penelitian ini merupakan asumsi yang didasarkan pada beberapa sumber sebagaimana ditunjukkan pada Tabel 4.1 dan Tabel 4.2. Kelainan pada tanaman melon yang berhubungan dengan daun dapat disebabkan oleh hama/penyakit atau kekurangan nutrisi. Tabel 4.1 menyajikan kelainan pada tanaman melon yang disebabkan oleh hama/penyakit berdasarkan situs resmi Seminis yaitu seminis-us.com, sebuah *brand* di bidang pertanian milik perusahaan agrokimia Mosanto yang telah diakuisisi oleh perusahaan farmasi Bayer. Tabel 4.2 menunjukkan defisiensi nutrisi pada tanaman melon berdasarkan situs Yara yaitu yara.us, sebuah industri pertanian yang berfokus pada nutrisi tanaman, solusi lingkungan, dan solusi pertanian digital.

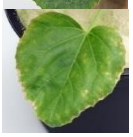
Untuk mempermudah deteksi, anotasi diprioritaskan pada citra objek daun yang terdekat dengan kamera, yakni daun pada tanaman yang terletak pada media tanam di baris terdekat dengan kamera. Adapun tanaman di baris kedua terdekat adalah opsional atau prioritas kedua untuk deteksi, sedangkan tanaman yang terletak lebih jauh dari itu tidak diberi label. Karena jumlah daun yang dideteksi sangat banyak dan terkadang saling menutupi atau bertumpuk, maka daun yang dideteksi hanya yang dapat terlihat bagian depannya.

Tabel 4.1 Hama/penyakit pada tanaman melon

No	Hama/Penyakit	Gejala/Tanda	Contoh
1	*Powdery Mildew*	Warna putih seperti tepung di bagian permukaan atas dan bawah daun, serta pada tangkai daun dan batang.	
2	*Cucurbit Yellow Stunting Disorder*	Daun berbintik diikuti klorosis di antara tulang daun (*interveinal chlorosis*) dan daun menguning. Seiring waktu, daun akan mulai menggulung ke atas dan menjadi rapuh.	
3	*Cucumber Green Mottle Mosaic*	Bercak hijau tua pada daun. Daun muda mungkin menunjukkan perubahan bentuk vena (*vein*) daun yang kusut.	

Tabel 4.2 Defisiensi nutrisi pada tanaman melon

No	Defisiensi Nutrisi	Gejala/Tanda	Contoh
1	Boron	Daun muda lebih kecil dari biasanya dan mungkin melengkung. Daun menguning dari daerah marginal antara vena ke arah tengah.	

No	Defisiensi Nutrisi	Gejala/Tanda	Contoh
2	Fosfor	Daun lebih kecil dengan warna hijau tua kusam. Mungkin terlihat perubahan warna ungu pada bagian bawah daun. Tanaman muda yang terkena parah menunjukkan bercak kecoklatan dan kemudian pembusukan total daun tua.	
3	Kalsium	Daun mungkin terlihat seperti kerangka dengan vena yang terhambat dan tidak berkembang dengan baik. Mulai dari tepi daun, klorosis menyebar ke seluruh helai daun. Daun lebih tua terlihat agak berbintik, sedangkan daun yang lebih muda dan lebih parah menjadi kuningan dengan vena hijau.	
4	Mangan	Daun menunjukkan binti *interveinal chlorosis*. Terdapat bercak ucat berair dengan batas coklat halus, terutama pada tepi daun.	
5	Magnesium	*Interveinal chlorosis* yang bisa menjadi keputihan dan akhirnya nekrotik. Vena daun tetap hijau.	
6	Molibdenum	Klorosis interveinal dan marginal dengan tepi hangus. Bintik keputihan mungkin muncul di helai daun. Daun kerdil dan cacat. Jika parah, tepi daun akan melengkung dan daun mati.	
7	Nitrogen	Daun tetap kecil dan berubah menjadi hijau pucat.	
8	Potasium	Daun muda bergelombang. Daun lebih tua menunjukkan tepi daun nekrotik, yang disebut ‘daun hangus’, yang khas untuk defisiensi K. Bintik coklat menyebar dari sana ke arah tengah.	
9	Sulfur	Pertumbuhan tanaman berkurang, daun menunjukkan warna hijau pucat kekuningan. Daun yang lebih tua mungkin menunjukkan bintik-bintik kecil dan cekung di seluruh helai daun dan bercak coklat muda di tepi.	

Untuk daun yang termasuk kelas normal harus terlihat secara keseluruhan, yakni tidak tertutupi oleh objek lain kecuali hanya sebagian kecil sehingga daun tetap dapat dipastikan sebagai daun normal. Adapun daun yang abnormal dapat dikategorikan pada kelas abnormal meskipun tertutupi objek lain selama dapat dipastikan sebagai daun yang abnormal.

Proses anotasi menggunakan labelImg menghasilkan *file* dengan format *.xml. Untuk pelatihan model Faster R-CNN dan SSD, *file* *.xml kemudian

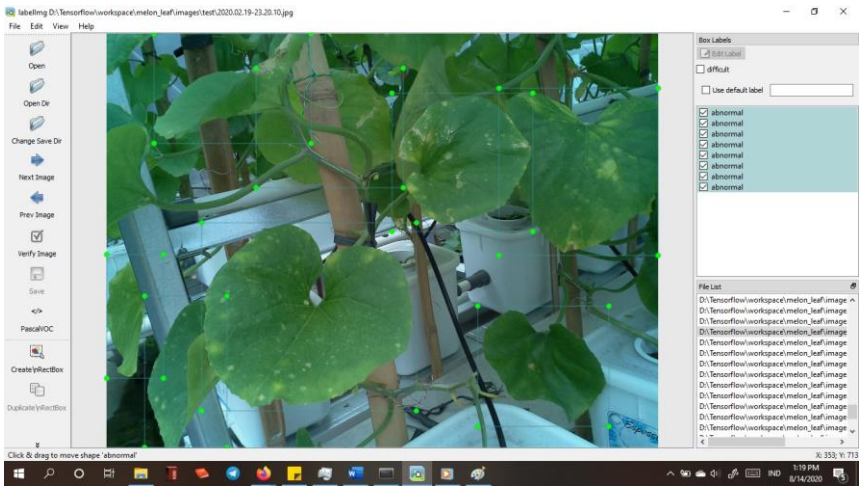
dikonversi ke dalam *file* *.csv yang memuat data empat koordinat (x_min, y_min, x_max, dan y_max) dan kelas label (abnormal atau normal) seperti pada Tabel 4.3. Setelah itu, *file* *.csv beserta dataset citra dikonversi ke dalam *file* TFRECORD dengan format *.record. *File* *.record inilah yang menjadi *input* untuk pelatihan model *object detection*.

(a) (b)

Gambar 4.2 Contoh daun (a) abnormal dan (b) normal

Untuk pelatihan model YOLOv3, *file* *.xml dikonversi ke dalam *file* dengan format *.txt. Setiap baris dalam *file* *.txt tersebut berisi data tentang citra dengan format *indeks_gambar path_ke_gambar lebar_gambar tinggi_gambar box_1 box_2 ... box_n*. Setiap *box_x* sebanyak n *box* ditulis dengan format *indeks_label x_min y_min x_max y_max*. Variabel *indeks_gambar* dan *indeks_label* berisi indeks angka yang dimulai dari nol (0). Contoh hasil konversi ke dalam format *.txt yaitu sebagai berikut.

```
0 /path_to_image/10.jpg 2592 1944 0 923 244 1460 722 1 333 1116
942 1563
1 /path_to_image/100.jpg 2592 1944 0 1655 1319 2425 1944 0 1570
1037 1789 1145
2 /path_to_image/106.jpg 2592 1944 0 289 889 775 1310 0 1366 786
1769 1177
```



Gambar 4.3 Proses anotasi data menggunakan labelImg

Tabel 4.3 Contoh konten *file* CSV

filename	*width*	*height*	*class*	*xmin*	*ymin*	*xmax*	*ymax*
103.jpg	2592	1944	normal	914	1039	1503	1560
103.jpg	2592	1944	normal	2	668	911	1243
103.jpg	2592	1944	abnormal	288	1498	717	1810

4.3 Pelatihan Model

Dataset dibagi secara acak sebanyak 80% untuk pelatihan dan 20% untuk pengujian. Dengan demikian, jumlah data pelatihan dan pengujian berturut-turut adalah 417 dan 105. Selanjutnya pelatihan model dilakukan menggunakan *framework* Tensorflow 1.0. *Source code* yang digunakan pada penelitian ini telah tersedia pada https://github.com/wizyoung/YOLOv3_TensorFlow untuk YOLOv3, sedangkan untuk Faster R-CNN dan SSD digunakan Tensorflow Object Detection API yang tersedia pada https://github.com/tensorflow/models/tree/r1.13.0.

Pelatihan model *deep learning* dilakukan menggunakan GPU (*graphics processing unit*). Bekerja dengan GPU secara interaktif di HPC LIPI dapat dilakukan dengan mengikuti langkah-langkah pada Lampiran 7. Langkah-langkah instalasi *package* dan Tensorflow *Object Detection* API terdapat pada Lampiran 8. Adapun proses pelatihan model *deep learning* dapat dilihat pada Lampiran 9.

4.3.1 *Anchor/Default/Prior Box*

Anchor box awal untuk setiap algoritme ditentukan dengan metode yang berbeda-beda sebagaimana berikut.

a) Faster R-CNN

Penentuan *anchor* untuk Faster R-CNN pada pelatihan ini menggunakan nilai skala dan rasio aspek dengan perhitungan lebar *anchor* (*w*) dan tinggi *anchor* (*h*) sebagaimana pada Formula 12 dan 13. Dalam penelitian ini digunakan kombinasi dari 3 nilai skala dan rasio aspek yaitu (0,5; 1,0; 2,0) sehingga diperoleh 9 *anchor box* awal. Dengan nilai lebar × tinggi *anchor* basis adalah 256 × 256, maka ukuran *anchor box* untuk pelatihan Faster R-CNN ialah sebagaimana pada Tabel 4.4.

Tabel 4.4 Ukuran *anchor* untuk Faster R-CNN

Skala	Rasio Aspek	Ukuran *Anchor*	
		Tinggi	Lebar
0,5	0,5	181,0	90,5
0,5	1,0	128,0	128,0
0,5	2,0	90,5	181,0
1,0	0,5	362,0	181,0
1,0	1,0	256,0	256,0
1,0	2,0	181,0	362,0
2,0	0,5	724,0	362,0
2,0	1,0	512,0	512,0
2,0	2,0	362,0	724,0

$$w = skala \times \sqrt{rasio\ aspek} \times lebar\ anchor\ basis \quad (12)$$

$$h = skala/\sqrt{rasio\ aspek} \times tinggi\ anchor\ basis \quad (13)$$

b) SSD

Untuk melakukan prediksi dalam pelatihan SSD kali ini digunakan 6 lapis (*layer*) *feature map.* Skala minimum (s_{min}) yang digunakan adalah 0.2 dan skala maksimum (s_{max}) adalah 0,95. Dengan demikian, menggunakan Formula 3 dan 6 skala s_k dan skala interpolasi (s'_k) untuk masing-masing *feature map k* adalah sebagaimana pada Tabel 4.5.

Tabel 4.5 Skala untuk *default box* pada *feature map layers*

	Skala (s_k)	Skala interpolasi (s'_k)
Layer 1	0,20	0,26
Layer 2	0,35	0,42
Layer 3	0,50	0,57
Layer 4	0,65	0,72
Layer 5	0,80	0,87
Layer 6	0,95	1,02

Lima rasio aspek ditetapkan untuk *default box* yang didefinisikan sebagai $a_r \in \left\{1.2.3.\frac{1}{2}.\frac{1}{3}\right\}$. Dengan demikian, menggunakan Formula 4 dan 5 skala lebar (*w*) dan tinggi (*h*) masing-masing *default box* adalah sebagaimana pada Tabel 4.6.

Tabel 4.6 Skala lebar dan tinggi *default box*

Rasio aspek	*Layer* 1		*Layer* 2		*Layer* 3		*Layer* 4		*Layer* 5		*Layer* 6	
	w	*h*	*w*	*h*	*w*	*h*	*w*	*h*	*w*	*h*	*w*	*h*
1,0	0,20	0,20	0,35	0,35	0,50	0,50	0,65	0,65	0,80	0,80	0,95	0,95
2,0	0,28	0,14	0,49	0,25	0,71	0,35	0,92	0,46	1,13	0,57	1,34	0,67
3,0	0,35	0,12	0,61	0,20	0,87	0,29	1,13	0,38	1,39	0,46	1,65	0,55
0,5	0,14	0,28	0,25	0,49	0,35	0,71	0,46	0,92	0,57	1,13	0,67	1,34
0,33	0,12	0,35	0,20	0,61	0,29	0,87	0,38	1,13	0,46	1,39	0,55	1,65
1,0	0,26	0,26	0,42	0,42	0,57	0,57	0,72	0,72	0,87	0,87	1,02	1,02

c) YOLOv3

Prior box untuk YOLOv3 diperoleh dari hasil *k-means clustering* pada data *bounding box* pada data latih. Sebagaimana pada Formula 2, IOU digunakan sebagai fungsi jarak alih-alih fungsi Euclidean yang biasa digunakan pada *k-means*. Pada penelitian ini, jumlah *cluster* yang dipilih adalah 9 sehingga terdapat 9 *prior box* untuk melakukan prediksi. Dengan menjalankan skrip *.py sebagaimana pada Lampiran 1, diperoleh 9 *cluster* untuk dataset yang digunakan dalam penelitian ini yaitu: (18 × 14), (32 × 21), (39 × 36), (54 × 28), (57 × 51), (84 × 42), (78 × 73), (106 × 66), (129 × 109).

4.3.2 Augmentasi Data

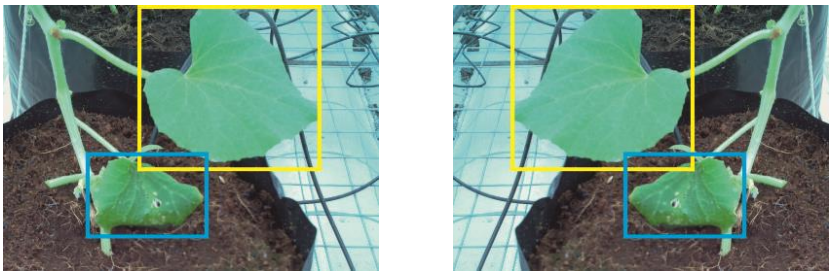
Terdapat dua metode melakukan teknik augmentasi saat pelatihan model, yaitu secara *offline* dan *online* (*on-the-fly*). Metode augmentasi secara *offline* dilakukan terpisah dari proses pelatihan. Teknik augmentasi terlebih dahulu diterapkan pada data citra, kemudian citra yang sudah diaugmentasi disimpan pada penyimpanan dan menjadi data latih tambahan yang baru. Adapun teknik augmentasi secara *online* atau *on-the-fly* dilakukan seiring berjalannya pelatihan model. Metode secara *online* lebih banyak digunakan karena lebih efisien dan tidak mengurangi kapasitas penyimpanan (*storage*) karena tidak perlu menyimpan data citra hasil augmentasi.

Pada penelitian ini teknik augmentasi *horizontal flip* diterapkan pada pelatihan seluruh model (Faster R-CNN, SSD, dan YOLOv3). Untuk pelatihan model Faster R-CNN dan SSD digunakan metode augmentasi secara *online* karena *Object Detection* API dari Tensorflow telah menyediakan modul/fungsi untuk melakukan *task* tersebut. Adapun untuk melatih model YOLOv3, dipilih metode augmentasi secara *offline* karena belum tersedia modul/fungsi untuk melakukan augmentasi secara *online*. Bagaimanapun, pada prinsipnya metode *offline* dan *online* menghasilkan hasil yang sama.

Untuk melakukan teknik augmentasi data, terlebih dahulu ditentukan sebanyak n_{aug} jumlah data yang akan diaugmentasi. Kemudian dari data latih dipilih sebanyak n_{aug} citra secara acak, lalu masing-masing data citra dibalik (*flip*) secara horizontal serta *bounding box* dan kelasnya disesuaikan dengan posisi citra yang baru. Adapun algoritme penentuan n_{aug} data augmentasi dengan probabilitas 0,5 adalah sebagai berikut.

While counter $<$ $\mathrm{n_{train}}$ **do**
  If $X \sim U(0.1) < 0{,}5$
    $n_{aug} = n_{aug} + 1$
  Else
    $n_{aug} = n_{aug}$
  counter = counter + 1

Dari proses perhitungan tersebut, dapat diperoleh jumlah data augmentasi sebanyak ~ 200. Gambar 4.4 menampilkan ilustrasi data citra sebelum dan sesudah dilakukan augmentasi *horizontal flip*.



(a) (b)

Gambar 4.4 Contoh data citra (a) sebelum dilakukan augmentasi dan (b) setelah dilakukan augmentasi *horizontal flip*

4.3.3 Konfigurasi Parameter

Beberapa konfigurasi parameter untuk pelatihan dilakukan sebagaimana pada Tabel 4.7. Pelatihan dilakukan dengan augmentasi data karena dapat menambah jumlah data latih sehingga membantu meningkatkan akurasi (Zoph *et al*. 2019). Teknik augmentasi yang digunakan adalah *random horizontal flip*. Adapun konfigurasi *pipeline* pelatihan dapat dilihat pada Lampiran 9.

Tabel 4.7 Konfigurasi parameter

Algoritme	*Backbone*	Ukuran *input*	*Batch size*	Jumlah *steps*
Faster R-CNN	InceptionV2 dan ResNet50	Min : 600 Maks : 1024	1	10.000
SSD	InceptionV2 dan MobileNetV2	300 × 300	16	10.000
YOLOv3	Darknet53	416 × 416	6	~10.000

Dalam menentukan nilai parameter tidak ada aturan baku, sehingga hal ini masih menjadi kekurangan dari metode *deep learning* karena harus mencoba berbagai nilai-nilai parameter yang berbeda yang membuatnya jadi tidak efisien. Terkadang satu peneliti berbeda dengan peneliti lainnya dalam menentukan parameter sesuai dengan pertimbangan masing-masing, bahkan tidak jarang peneliti tidak menjelaskan pertimbangan saat menentukan nilai parameter. Demikian halnya nilai parameter yang sama tidak selalu memberikan hasil yang sama ketika diimplementasikan pada dataset yang berbeda. Pada penelitian ini, nilai parameter ukuran *input* disesuaikan dengan ukuran yang digunakan pada penelitian sumber masing-masing algoritme. Pada Ren *et. al.* (2017) digunakan ukuran *input* 600 piksel untuk tepi yang lebih pendek, sedangkan tepi yang lebih panjang ukurannya disesuaikan berdasarkan skala tepi pendek. Misalnya, ukuran gambar asli adalah 800 × 1000 piksel, maka *input* untuk Faster R-CNN menjadi 600 × 750 piksel.

Adapun pada penelitian ini digunakan parameter dimensi *input* minimal yaitu 600 piksel dan dimensi maksimal yaitu 1024 piksel karena ukuran gambar yang terlalu besar. Ini berarti setiap gambar akan diubah ukurannya sedemikian hingga ukuran dimensi minimal 600 piksel sebagaimana pada Ren *et. al.* (2017). Adapun jika dimensi maksimal lebih dari 1024 piksel, maka ukuran gambar diubah sedemikian hingga ukuran dimensi maksimal 1024 piksel. Misalnya, hasil tangkapan kamera Raspberry Pi adalah 2592 × 1 944 piksel, maka ukuran *input* untuk Faster R-CNN menjadi 1024 × 768.

Ukuran *input* SSD ditentukan berdasarkan penelitian Liu *et. al.* (2016) yaitu 300 × 300. Liu *et. al.* (2016) melakukan percobaan dengan ukuran *input* 300 x 300 dan 512 x 512. Sebagaimana biasanya, ukuran *input* yang lebih besar memberikan hasil akurasi yang lebih baik karena informasi yang diperoleh dari gambar lebih banyak, meskipun menyebabkan waktu komputasi menjadi lebih lama. Adapun untuk penelitian ini digunakan ukuran 300 x 300 karena menghasilkan kinerja yang lebih cepat untuk implementasi pada perangkat komputasi terbatas. Adapun YOLOv3 menggunakan ukuran *input* 416 x 416 sebagaimana pada percobaan Redmon dan Farhadi (2018).

Nilai *batch size* atau *mini-batch size* dapat ditentukan mulai dari 1 dan seterusnya. Biasanya, nilai *batch size* dipilih > 1 menggunakan bilangan terdekat dari pangkat 2 seperti 16, 32, 64. Penentuan *batch size* biasanya berpengaruh pada waktu pelatihan ketika beberapa peneliti menggunakan *threshold* sebagai *stop condition*. Nilai *batch size* yang lebih besar biasanya menyebabkan *training loss* lama menuju konvergen meskipun lebih stabil saat konvergen karena data yang diproses pada satu kali langkah (*step*) lebih banyak. Penentuan nilai *batch size* menunjukkan seberapa banyak data latih yang *fit* pada GPU atau memori utama saat proses pelatihan. Faster R-CNN menggunakan *batch size* 1 karena pelatihan dilakukan dengan 1 GPU, sehingga komputasi disesuaikan dengan kemampuan GPU. Pelatihan model SSD dilakukan dengan *batch size* 16 agar lebih stabil saat konvergen, sedangkan YOLOv3 menggunakan *batch size* 6 saat pelatihan agar menuju konvergen lebih cepat. Adapun *stop condition* pada penelitian ini adalah jumlah *step* yaitu 10.000. Nilai *loss* pada saat pelatihan biasanya mulai konvergen dan tidak ada perubahan yang signifikan setelah sekitar *step* ke-7000, sehingga nilai 10.000 dianggap cukup sebagai *threshold*.

Ketika pelatihan model telah selesai, akan diperoleh *file checkpoint* yang merupakan jejak proses pelatihan. Terdapat tiga *file* utama yang tersimpan dengan tiga format yang berbeda, yaitu *.data-00000-of-00001, *.index, dan *.meta. Untuk melakukan *object detection* dengan model yang telah dilatih, *file* baru yaitu *trained inference graph* perlu diekstrak dari *checkpoint* hasil pelatihan. Hasil ekstraksi *trained inference graph* kemudian disimpan ke dalam *file* `frozen_inference_graph.pb` (Lampiran 9). Dalam penelitian ini proses pembuatan *trained inference graph* dilakukan untuk model Faster R-CNN dan SSD, sedangkan untuk YOLOv3 tidak perlu membuat *inference graph*. Proses ekstraksi *inference graph* dilakukan di perangkat yang sama di mana pelatihan model dilakukan (HPC) karena selanjutnya digunakan untuk melakukan *object detection task* untuk evaluasi model di perangkat tersebut. Adapun rincian ukuran *file* yang diperoleh dari hasil pelatihan dapat dilihat pada Lampiran 10.

4.4 Evaluasi Model

Evaluasi model *object detection* dilakukan dengan menghitung nilai *mean average precision* (mAP). Pada penelitian ini, penghitungan mAP dilakukan dengan nilai ambang batas IOU (IOU *threshold*) untuk proses *non-max suppression* (NMS) adalah 0,6. Berdasarkan Tabel 4.8, Faster R-CNN memiliki nilai mAP tertinggi dibandingkan dengan metode lainnya, sedangkan mAP terendah dimiliki oleh metode YOLOv3. Model Faster R-CNN dengan *backbone* ResNet50 menjadi model yang paling akurat dalam mendeteksi daun melon beserta label kelasnya dengan mAP 48,85%. Hal ini menunjukkan bahwa model deteksi *two-stage* lebih unggul dalam hal akurasi daripada model *one-stage*. Sebagaimana ditampilkan pada Gambar 4.5, metode deteksi *two-stage* memiliki keunggulan dalam mAP mencapai dua kali lipat dibandingkan dengan metode *one-stage*.

Sebagaimana pada Lu *et al*. (2020), beberapa kelebihan metode *two stage* dibandingkan dengan metode *one stage* di antaranya adalah sebagai berikut.

a. Dengan adanya *sampling* terhadap *region proposal*, *two-stage detectors* memfilter sebagian besar proposal negatif, sedangkan *one-stage detectors*

memproses semua *region* pada citra dan memiliki masalah ketidakseimbangan kelas (*class imbalance*).

b. Karena metode *two stage* hanya memproses sedikit proposal (dengan *sampling* terhadap *region proposal*), jaringan untuk klasifikasi dan regresi menjadi lebih besar, sehingga dapat mengekstrak lebih banyak fitur.
c. Fitur berkualitas tinggi dari proposal dapat diperoleh pada *two-stage detectors* dengan menggunakan operasi RoI Align yang mengekstrak fitur konsisten lokasi untuk setiap proposal.
d. *Two-stage detectors* melakukan regresi untuk lokasi objek dua kali (sekali di setiap *stage*) dan *bounding box* lebih baik daripada metode *one stage*.

Membedakan daun abnormal dan normal dari sekian banyak daun serta menentukan lokasinya merupakan permasalahan yang cukup kompleks. Tidak hanya perlu menentukan objek daun, tetapi juga membedakan kelasnya dan menentukan lokasi daun mana saja yang perlu dideteksi. Bagian *background* dapat terlihat mirip dengan objek positif karena pada citra sekumpulan daun terdapat objek daun baik di *background* maupun *foreground*. Sebagaimana pada proses anotasi data, tidak semua daun diberi label. Selain itu, daun yang bertumpuk menambah kepadatan objek deteksi. Dengan karakteristik dataset seperti itu, metode *two-stage* yang *powerful* dalam aspek kekayaan fitur yang diekstrak dapat memiliki akurasi lebih unggul daripada metode *one-stage*.

Tabel 4.8 Nilai AP dan mAP masing-masing metode

Metode	*Backbone*	$AP_{abnormal}$ (%)	AP_{normal} (%)	mAP (%)
Faster R-CNN	InceptionV2	**47,26**	50,2	48,73
	ResNet50	45,83	**51,86**	**48,85**
SSD	InceptionV2	23,83	30,25	27,04
	MobileNetV2	30,52	35,81	33,16
YOLOv3	Darknet53	15,26	17,85	16,56

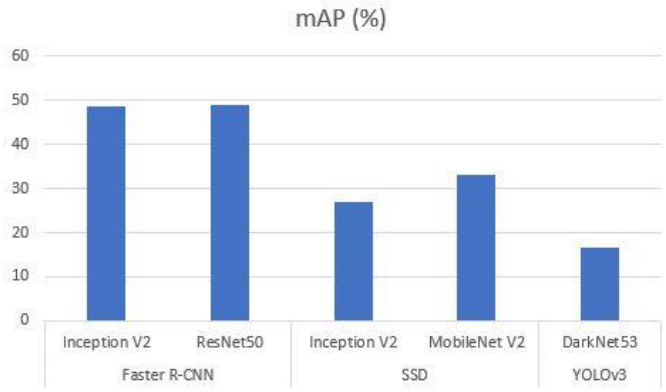
Model SSD menggunakan MobileNetV2 menghasilkan nilai mAP yang lebih tinggi daripada menggunakan InceptionV2 dengan selisih ~6%. MobileNetV2 mungkin lebih baik daripada InceptionV2 dalam ekstraksi fitur. *Inverted Residual Block* (Sandler *et al*. 2019) bisa jadi memiliki peran dalam memperkaya fitur dari citra dengan karakteristik yang bertumpuk dan kompleks seperti kelainan daun.

Adapun YOLOv3 menghasilkan nilai mAP yang paling kecil dibandingkan dengan metode lainnya yaitu 16,56%. Nilai mAP ini mencapai 2× lebih kecil daripada SSD MobileNetV2. Untuk kasus yang membutuhkan tingkat akurasi tinggi seperti kelainan daun, model YOLOv3 tidak direkomendasikan untuk digunakan karena akan menghasilkan lebih banyak kesalahan deteksi. Sepertinya YOLOv3 tidak begitu baik dalam mengambil informasi (mengekstrak fitur) dari data yang kompleks dan bertumpuk atau padat seperti daun. Hal ini bisa jadi karena *prior box* di setiap skala bekerja kurang baik ketika mengambil informasi. Tidak seperti SSD yang menggunakan enam *default box* di setiap *featur map layer*, YOLOv3 menggunakan lebih sedikit (tiga) *prior box* di setiap skala untuk mendeteksi objek sehingga tidak dapat menggali informasi lebih banyak.

Pada data daun, latar belakang (*background*) terlihat mirip dengan objek positif yang membuatnya menjadi kompleks terutama untuk *one-stage detectors*. Selain itu, YOLOv3 termasuk pada algoritme yang kurang baik dengan operasi

NMS pada area objek yang padat, sehingga memiliki tingkat kesalahan lebih tinggi pada saat mendeteksi objek yang padat (Lei *et al*. 2019). Pada penelitan ini, YOLOv3 menghasilkan banyak deteksi ganda pada objek yang sama sebagaimana dapat dilihat di hasil deteksi. Pada penelitian Nguyen *et al*. (2020) YOLOv3 mengalami penuruan mAP secara drastis ketika IOU dinaikkan dari 0,5 menjadi 0,75 karena YOLOv3 tidak berkerja dengan baik pada saat lokalisasi.

Untuk meningkatkan akurasi, di antaranya dapat dilakukan dengan memodifikasi struktur jaringan suatu algoritme sesuai dengan karakteristik dataset. Citra dengan objek yang berukuran kecil atau padat dan bertumpuk membutuhkan algoritme yang menghasilkan fitur ekstraksi lebih kaya. Sehingga, algoritme dapat disesuaikan agar dapat mengekstrak fitur lebih kaya dari citra yang memiliki karakteristik seperti itu. Misalnya, Huang *et al*. (2020) menambahkan jaringan modul SPP (*Spatial Pyramid Pooling*) pada YOLOv3 untuk memperbaiki informasi yang diperoleh dari fitur abstrak yang disebabkan oleh jaringan ekstraksi fitur yang semakin dalam. Hal itu bertujuan untuk meningkatkan kinerja YOLOv3 dalam mendeteksi objek kecil. Li *et al*. (2019) menggunakan 12 *prior box* pada YOLOv3 dan memperoleh mAP sekitar 20% lebih tinggi daripada menggunakan 9 *prior box*. Selain itu, Arsenovic *et al*. (2019) menambahkan jaringan PDNet (PlantDiseaseNet) untuk memperbaiki kinerja Faster R-CNN, YOLOv3, SSD, dan RetineNet dalam mendeteksi penyakit tanaman menggunakan dataset PlantDisease.



Gambar 4.5 Perbandingan mAP masing-masing algoritme

4.5 Implementasi pada Raspberry Pi

Pembuatan *inference graph* harus dilakukan pada perangkat di mana akan dilakukan *object detection task*. Oleh karena itu, selain membuat *file inference graph* pada perangkat komputer di mana proses pelatihan berlangsung, dilakukan juga proses ekstraksi yang sama di perangkat Raspberry Pi.

Object detection task dilakukan pada perangkat Raspberry Pi bertujuan untuk menganalisis kemampuan perangkat tersebut dalam menjalankan algoritme *object detection*. Pada tahap implementasi model, diukur waktu komputasi algoritme dalam melakukan inferensi deteksi yang selanjutnya disebut dengan *inference time*. Sebagaimana dapat dipahami dari situs resmi Tensorflow, yaitu tensorflow.org, tahap inferensi berarti tahap di mana model menentukan atau mengambil kesimpulan tentang *bounding box* dan label kelas pada setiap objek yang terdeteksi oleh model pada suatu gambar. Selain itu, pada implementasi algotime ini

dilakukan pengukuran terhadap penggunaan sumber daya, yakni seberapa banyak algoritme menghabiskan sumber daya seperti memori dan CPU dalam melakukan proses *object detection*.

Proses deteksi objek dilakukan terhadap 10 data citra yang dipilih secara acak dari data uji. Untuk setiap gambar dilakukan proses deteksi dan diukur waktu komputasi pada saat algoritme melakukan inferensi. Selanjutnya, dihitung rata-rata waktu komputasi dari data *inference time* yang diperoleh. Pengambilan 10 data citra bertujuan untuk mempersingkat proses pengukuran waktu komputasi karena jika digunakan seluruh data uji pada Raspberry Pi akan memakan waktu yang jauh lebih lama daripada saat menghitung mAP menggunakan GPU.

Selanjutnya dilakukan proses deteksi terhadap satu data uji untuk mengukur seberapa banyak sumber daya yang digunakan oleh setiap algoritme. Parameter yang diukur meliputi penggunaan CPU (*Central Processing Unit*) dan RSS (*Resident Set Size*). Penggunaan sumber daya dimonitor selama program berjalan.

4.6 Analisis Perbandingan Kinerja Algoritme

Metode *two-stage* memiliki keunggulan daripada metode *one-stage* dalam hal akurasi, tetapi metode *two-stage* memiliki kelemahan dalam hal waktu komputasi. Pada penelitian ini, waktu komputasi yang dibutuhkan oleh metode *two-stage* (Faster R-CNN) untuk melakukan deteksi lebih lama daripada metode *one-stage* (SSD dan YOLOv3). Metode *two stage* melakukan tahap *region proposal* dan klasifikasi dan regresi secara terpisah sehingga menambah waktu komputasi. Di setiap tahap, metode *two stage* melakukan klasifikasi, yang pertama yaitu untuk menentukan keberadaan objek dan yang kedua untuk menentukan label kelas.

Tabel 4.9 menampilkan perbandingan mAP dan *inference time* masing-masing metode dengan Faster R-CNN ResNet50 sebagai *benchmark*. Untuk metode *one-stage*, waktu komputasi tercepat dimiliki oleh YOLOv3 dengan kecepatan komputasi 0,5 detik, yakni mencapai lima kali lebih cepat daripada metode Faster R-CNN dan SSD MobileNetV2 sebagaimana ditunjukkan pada Tabel 4.9. Meskipun YOLOv3 menjadi yang tercepat, tetapi nilai mAP yang dimiliki masih sangat kecil bahkan di bawah 20%. Sehingga metode YOLOv3 tidak direkomendasikan untuk mendeteksi kelainan daun tanaman melon.

SSD dengan InceptionV2 membutuhkan waktu komputasi 1,2 kali lebih cepat daripada SSD dengan MobileNetV2 tetapi memiliki mAP ~6% lebih kecil. Karena pada kasus deteksi kelainan daun akurasi lebih diutamakan, maka dapat dikatakan bahwa SSD MobileNetV2 lebih baik daripada SSD InceptionV2 karena memiliki mAP yang lebih tinggi dengan waktu komputasi yang masih dapat dipertimbangkan untuk diterapkan pada perangkat komputasi terbatas.

Faster R-CNN membutuhkan waktu komputasi yang paling lama dibandingkan dengan algoritme deteksi lainnya. Dibandingkan dengan YOLOv3, Faster R-CNN melakukan inferensi lima kali lebih lama, tetapi memiliki nilai mAP mencapai tiga kali lebih besar. Kedua algoritme tersebut paling unggul di satu aspek dan lemah pada aspek lainnya, tetapi pada penelitian ini akurasi lebih diutamakan daripada waktu komputasi. Faster R-CNN ResNet50 memiliki nilai mAP yang paling tinggi dengan waktu komputasi sedikit lebih lama daripada SSD MobileNetV2 yaitu sekitar 1,2 kali lebih lama. Faster R-CNN dengan ResNet50 dan InceptionV2 memiliki kinerja yang hampir sama dengan mAP ~49% dan *inference time* ~2,5 detik, namun jika skala yang lebih kecil diperhitungkan maka

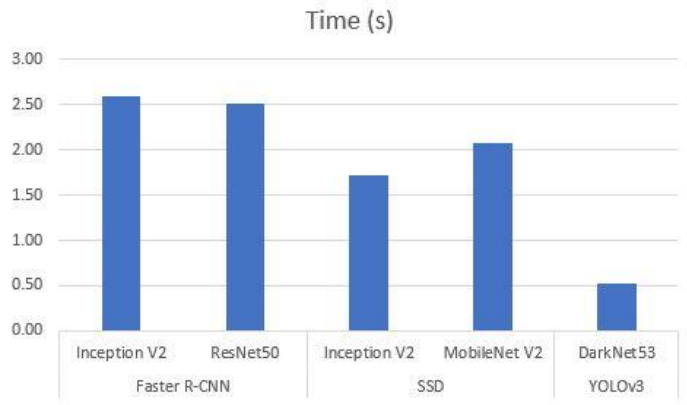
Faster R-CNN ResNet50 lebih baik karena membutuhkan waktu 84 milidetik lebih cepat. Dalam hal ini, mungkin *residual block* pada jaringan ResNet50 memiliki peran dalam menaikkan akurasi (He *et al*. 2015).

Dari uraian tersebut dapat disimpulkan bahwa metode Faster R-CNN dapat dipertimbangkan untuk diterapkan pada perangkat komputasi terbatas seperti Raspberry Pi karena memiliki tingkat akurasi yang tinggi dengan waktu komputasi yang masih dapat dipertimbangkan. Jika diinginkan proses deteksi objek secara *realtime* seperti untuk video, maka metode deteksi *one-stage* seperti SSD lebih direkomendasikan.

Gambar 4.7 menampilkan *scatter plot* mAP terhadap waktu komputasi algoritme *object detection*. Gambar 4.5 menunjukkan bahwa waktu komputasi berbanding lurus dengan mAP, yakni semakin tinggi nilai mAP maka waktu komputasi yang dibutuhkan juga semakin lama. Faster R-CNN ResNet50 menjadi model dengan mAP terbaik yang sedikit lebih cepat daripada Faster R-CNN InceptionV2. Faster R-CNN ResNet50 sangat direkomendasikan untuk kasus deteksi kelainan daun tanaman melon jika pengiriman dan pemrosesan data, misalnya oleh robot pengawas, dilakukan dalam waktu tidak kurang dari lima menit. Hal ini tidak menjadi masalah karena tanda atau gejala kelainan pada daun tanaman masih dapat dideteksi meskipun tidak dalam hitungan detik. Adapun jika diinginkan pemrosesan yang lebih cepat seperti untuk video secara *realtime*, maka metode SSD lebih direkomendasikan.

Tabel 4.9 Nilai mAP dan *inference time* masing-masing algoritme

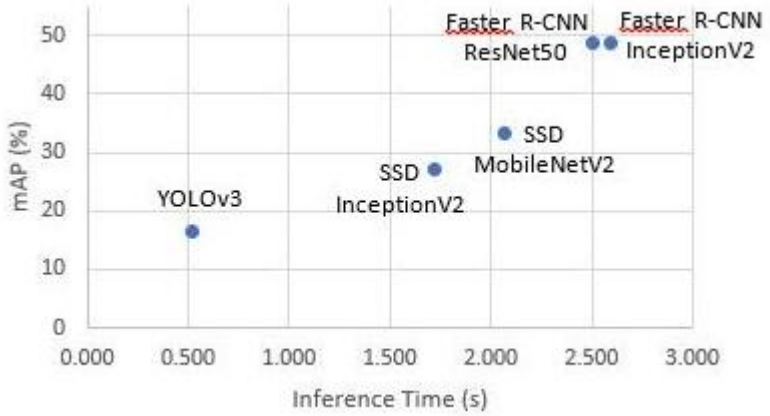
Metode	*Backbone*	mAP (%)	*Inference time* (s) per gambar (*Speed up*)
Faster R-CNN	InceptionV2	48,73	2,591 (1×)
	ResNet50	**48,85**	2,507 (1×)
SSD	InceptionV2	27,04	1,716 (1,5×)
	MobileNetV2	33,16	2,07 (1,2×)
YOLOv3	Darknet53	16,56	**0,522** (4,8×)



Gambar 4.6 Perbandingan *inference time* masing-masing algoritme

Secara umum, dapat disimpulkan bahwa untuk kasus yang membutuhkan akurasi tinggi, metode *two-stage* Faster R-CNN dapat menjadi alternatif terbaik dengan waktu komputasi yang masih dapat dipertimbangkan untuk penerapan pada perangkat komputasi terbatas seperti Raspberry Pi. Adapun untuk kebutuhan

deteksi secara *realtime* seperti pada video, SSD dapat menjadi pilihan alternatif terbaik untuk digunakan.



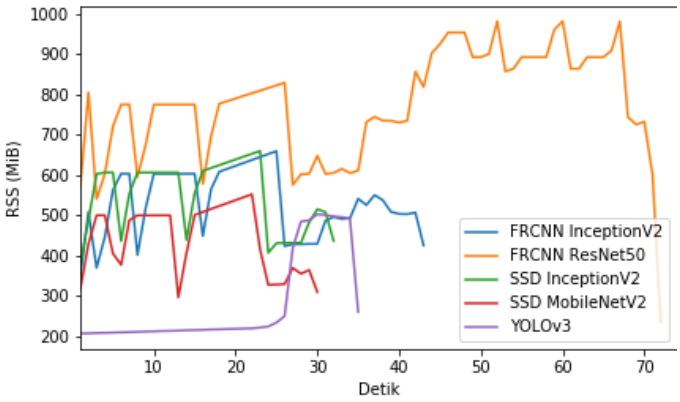
Gambar 4.7 *Scatter plot* mAP terhadap *inference time*

Gambar 4.8 menunjukkan jumlah memori (RAM) yang digunakan oleh program *object detection* ketika dijalankan. Sumbu *x* mewakili waktu selama program berjalan yaitu setiap detik, dimulai dari program dieksekusi sampai menghasilkan *output* deteksi objek, sedangkan sumbu *y* menunjukkan jumlah RSS (MiB) yang dialokasikan untuk menjalankan masing-masing program deteksi. RSS menunjukkan memori *non-swap* atau RAM yang digunakan oleh *task* (Kerrisk 2020). Secara umum, di detik-detik awal diperlihatkan penggunaan memori yang lebih dominan ditandai dengan pergerakan grafik RSS yang lebih fluktuatif daripada penggunaan CPU yang cenderung stabil. Pada tahap awal ini biasanya program sedang memuat modul atau *package* dan memuat model *object detection*. Proses pemuatan *package* dan model tidak terlalu membebani CPU karena tidak terdapat perintah untuk operasi perhitungan yang rumit sehingga kompleksitas tertuju pada memori.

Penggunaan memori terbesar dimiliki oleh Faster R-CNN ResNet50 yang mana maksimum penggunaan RSS mencapai 981 MiB. Hal ini wajar karena Faster R-CNN merupakan model *two stage* yang memiliki arsitektur kompleks (yakni dua tahap: *region proposal* dan klasifikasi) sehingga memengaruhi penggunaan memori lebih banyak, ditambah dengan ResNet50 sebagai *backbone* yang memiliki jaringan lebih dalam daripada InceptionV2 (He *et al*. 2015). Selain itu, proses deteksi Faster R-CNN ResNet50 membutuhkan lebih banyak memori sebagaimana terlihat pada Gambar 4.8 yang menunjukkan pergerakan grafik RSS di detik-detik akhir menjadi lebih tinggi.

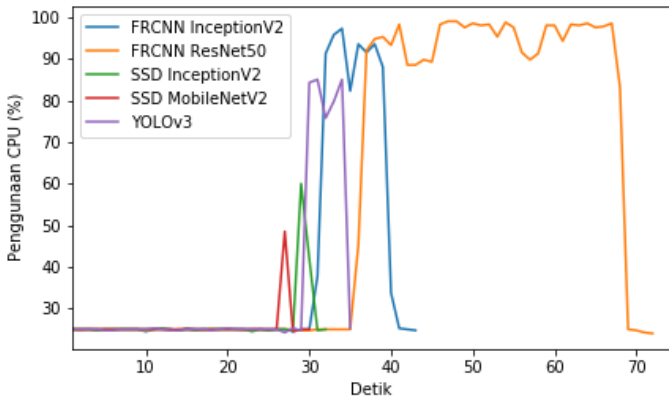
Penggunaan memori terkecil dimiliki oleh YOLOv3 dan SSD MobileNetV2 dengan maksimum penggunaan memori berturut-turut adalah 502 MiB dan 551 MiB. YOLOv3 menjadi lebih ringan bisa jadi karena struktur desain algoritme program lebih efisien daripada Faster R-CNN dan SSD yang menggunakan *Object Detection API*. Sebagaimana terlihat pada Gambar 4.6, pergerakan RSS YOLOv3 secara signifikan naik setelah detik ke-25 yang kemungkinan ketika program sedang melakukan proses deteksi, berbeda dengan grafik SSD dan Faster R-CNN yang memiliki pola fluktuasi RSS yang mirip di detik-detik awal (sebelum detik ke-25 atau ke-30). Selain itu, mungkin juga karena model YOLOv3 yang efisien sebagai model *one stage* meskipun dari aspek *backbone* ukuran Darknet53 bahkan

lebih besar daripada ResNet50 (Elgendi *et al.* 2020). Adapun SSD MobileNetV2 lebih ringan dimungkinkan karena model SSD yang efisien sebagai *one stage object detector* dan arsitektur MobileNetV2 dengan arsitektur *mobile* yang didesain untuk perangkat dengan kemampuan komputasi rendah (Sandler *et al*. 2019; Srivastava *et al*. 2019).



Gambar 4.8 Penggunaan memori

Gambar 4.9 menunjukkan persentase penggunaan CPU oleh program *object detection* ketika dijalankan. Persentase penggunaan CPU berarti persentase waktu CPU (CPU *time*) yang digunakan oleh *task* (Kerrisk 2020). Secara umum, setelah detik ke-25 penggunaan memori mulai menurun, kemudian diikuti dengan pergerakan grafik persentase CPU yang menuju puncak. Hal ini menandai proses deteksi sedang berjalan. Pada Raspberry Pi proses pengolahan citra dilakukan di CPU, sehingga pada tahap ini dilakukan berbagai perhitungan matematis oleh *deep learning* yang membebani CPU.



Gambar 4.9 Penggunaan CPU

Metode deteksi *two stage* serta *backbone* dengan jaringan yang dalam menjadikan Faster R-CNN ResNet50 memiliki CPU *time* paling tinggi dengan persentase penggunaan mencapai CPU 99% sebagaimana pada Tabel 4.10. SSD MobileNetV2 menjadi algoritme yang paling sedikit menghabiskan sumber daya CPU dengan persentase 48,5%. Di antara metode *one-stage*, YOLOv3 menghabiskan paling sedikit memori, tetapi memiliki CPU *time* maksimum 85%

yang mana lebih besar daripada metode SSD. Meskipun bekerja jauh lebih cepat, YOLOv3 membutuhkan spesifikasi sumber daya (seperti CPU atau GPU) yang lebih tinggi untuk menjalankan komputasi karena berkaitan dengan kompleksitas algoritme YOLOv3. Selain itu, pada kasus tertentu Darknet memiliki proses komputasi yang tidak terlalu dibutuhkan, sehingga memodifikasi arsitektur jaringan dapat menghasilkan kinerja algoritme yang lebih efisien (Mansoub *et al*. 2019). Mansoub *et al*. (2019) dan Mao *et al*. (2019) memodifikasi atau menyederhanakan algoritme YOLOv3 sehingga menjadi lebih efisien secara komputasi dengan tetap mempertahankan akurasi.

Dari analisis tersebut, dapat disimpulkan bahwa dari aspek penggunaan sumber daya, SSD MobileNetV2 adalah metode terbaik untuk diimplementasikan pada perangkat komputasi terbatas seperti Raspberry Pi. Selain memiliki mAP yang cukup, SSD MobileNetV2 menggunakan sumber daya lebih efisien daripada model lainnya. Agar memperoleh hasil yang lebih optimal baik dari aspek akurasi maupun beban komputasi, arsitektur jaringan suatu algoritme *deep learning* dapat dimodifikasi dan disesuaikan berdasarkan kebutuhan kasus, seperti berkaitan dengan karakteristik dataset dan perangkat keras yang digunakan.

Tabel 4.10 Penggunaan maksimum CPU dan RSS

Metode	*Backbone*	CPU (%)	RSS (MiB)
Faster R-CNN	InceptionV2	97,25	659
	ResNet50	99	981
SSD	InceptionV2	60	659
	MobileNetV2	**48,5**	551
YOLOv3	Darknet53	85	**502**

4.7 Hasil Deteksi

Bagian ini manampilkan contoh hasil deteksi algoritme-algoritme *object detection* yang dibahas pada penelitian ini terhadap sebuah citra daun tanaman melon. Gambar 4.10 adalah citra yang akan dideteksi objek daun beserta label kelasnya. Gambar 4.11 menunjukkan citra daun tanaman melon disertai *ground truth box*, yakni *box* beserta label kelas hasil anotasi data.

Gambar 4.12 sampai dengan Gambar 4.26 merupakan hasil deteksi yang dilakukan oleh berbagai algoritme *object detection* dengan beberapa nilai ambang batas IOU (IOU *threshold*) yang berbeda-beda untuk *non-max suppression* (NMS) yaitu 0,6; 0,4; 0,2. Nilai IOU *threshold* yang semakin kecil akan mengembalikan jumlah *bounding box* yang lebih sedikit. Percobaan ini dilakukan karena karakteristik objek daun yang bertumpuk dan saling berdekatan, sehingga beberapa *bounding box* hasil prediksi menunjukkan objek dan kelas yang sama dengan seperti pada Gambar 4.24 dan Gambar 4.25.

Sebagai gambaran untuk penggunaan *object detection* pada robot pengawasan, model *object detector* dapat diimplementasikan pada robot menggunakan Raspberry Pi. Robot dengan kamera terlebih dahulu dapat mengambil gambar sekumpulan daun tanaman melon. Kemudian dilakukan proses *object detection* pada sistem robot (Raspberry Pi) sampai mengeluarkan hasil deteksi. Jika hasil deteksi menunjukkan terdapat daun yang abnormal, maka robot dapat melakukan

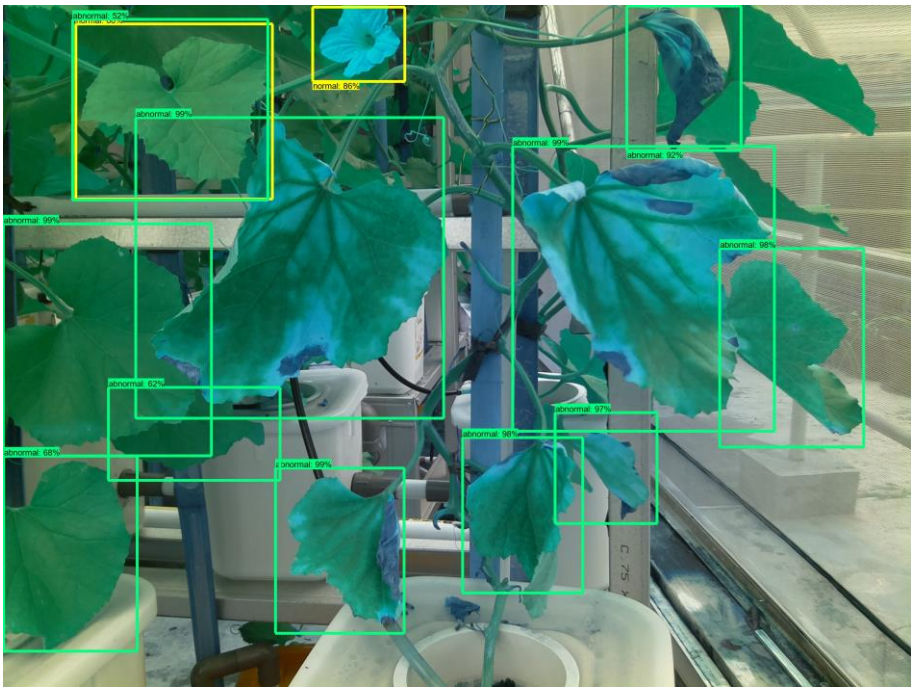
tindak lanjut seperti mengirimkan sinyal peringatan dini atau melakukan *prunning* terhadap daun yang abnormal. Robot tersebut juga dapat melakukan deteksi secara *realtime* menggunakan perangkat dan model *deep learning* yang sama. Di antara kelebihan penerapan sistem *edge computing* seperti ini dibandingkan dengan *cloud computing* adalah waktu server dan *bandwidth* lebih hemat karena tidak perlu mengirimkan data ke *cloud* dan tidak memerlukan koneksi internet. Selain itu, untuk mendukung penelitian ini telah dibuat aplikasi *object detection* untuk mendeteksi daun tanaman melon yang telah dicoba pada perangkat laptop sebagaimana dapat dilihat pada Lampiran 11.

Gambar 4.10 Citra untuk contoh deteksi

Gambar 4.11 Citra daun dengan *ground truth box*

Gambar 4.12 Contoh deteksi Faster R-CNN InceptionV2 (IOU untuk NMS = 0,6)

Gambar 4.13 Contoh deteksi Faster R-CNN InceptionV2 (IOU untuk NMS = 0,4)

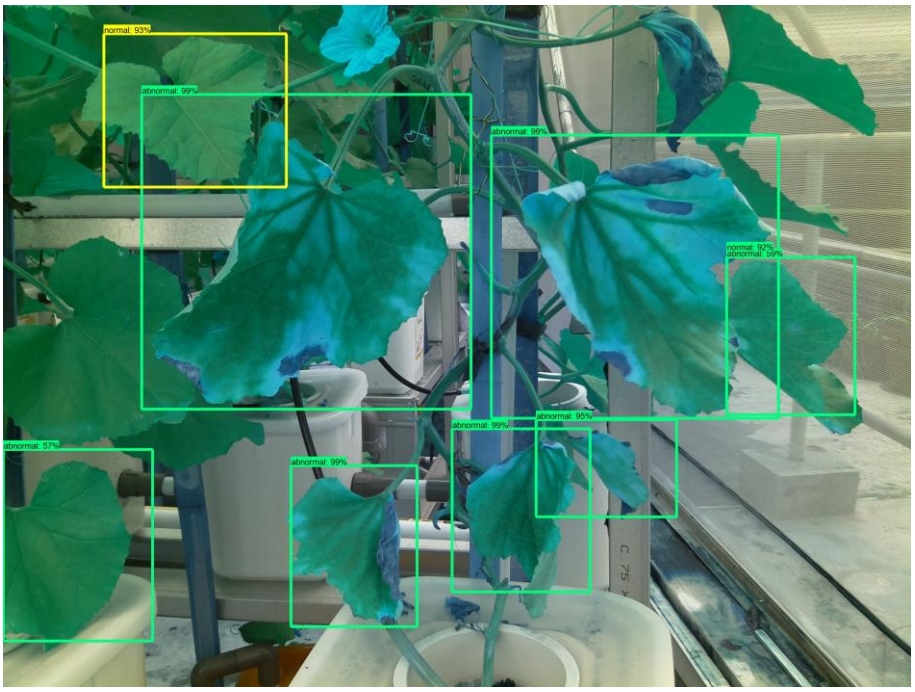


Gambar 4.14 Contoh deteksi Faster R-CNN InceptionV2 (IOU untuk NMS = 0,2)

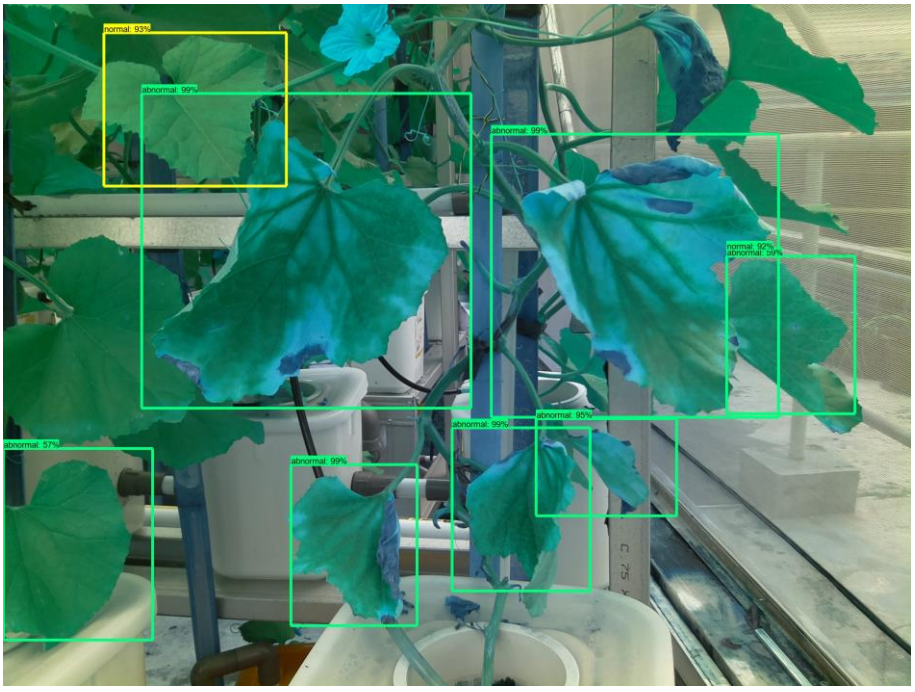
Gambar 4.15 Contoh deteksi Faster R-CNN ResNet50 (IOU untuk NMS = 0,6)

Gambar 4.16 Contoh deteksi Faster R-CNN ResNet50 (IOU untuk NMS = 0,4)

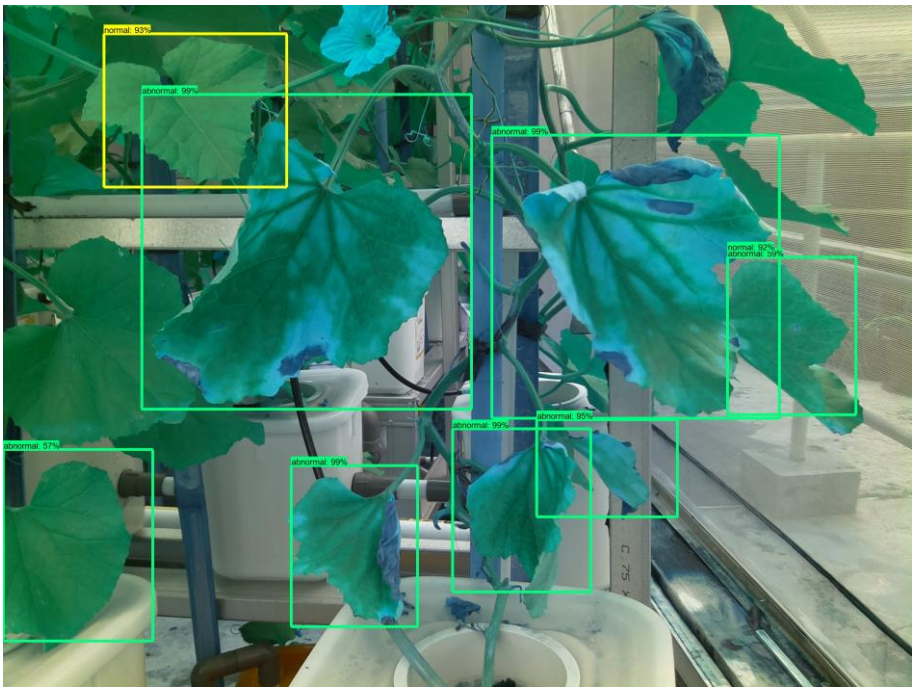
Gambar 4.17 Contoh deteksi Faster R-CNN ResNet50 (IOU untuk NMS = 0,2)



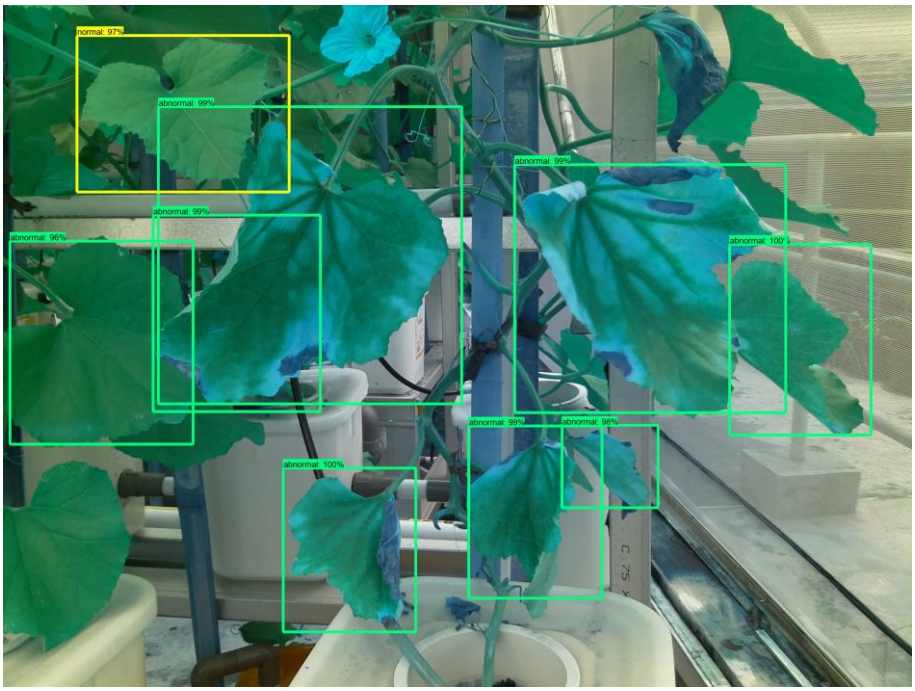
Gambar 4.18 Contoh deteksi SSD InceptionV2 (IOU untuk NMS = 0,6)



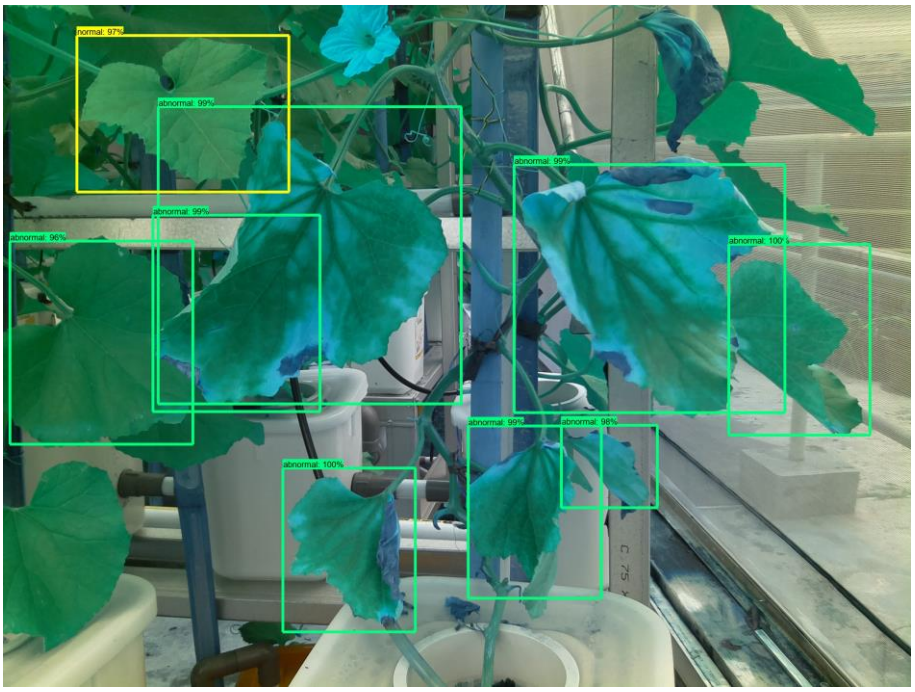
Gambar 4.19 Contoh deteksi SSD InceptionV2 (IOU untuk NMS = 0,4)



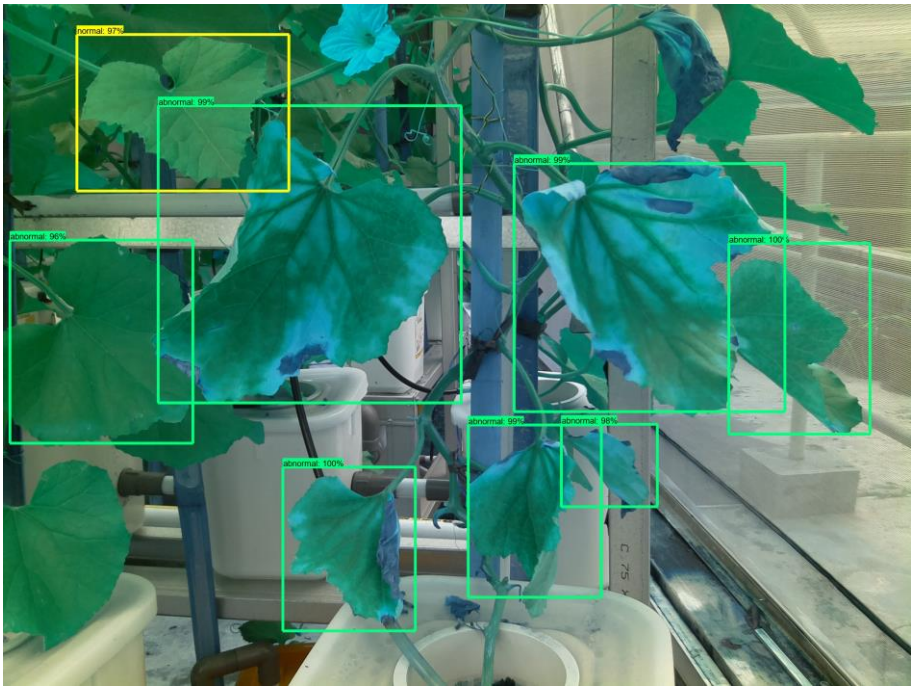
Gambar 4.20 Contoh deteksi SSD InceptionV2 (IOU untuk NMS = 0,2)



Gambar 4.21 Contoh deteksi SSD MobileNetV2 (IOU untuk NMS = 0,6)



Gambar 4.22 Contoh deteksi SSD MobileNetV2 (IOU untuk NMS = 0,4)



Gambar 4.23 Contoh deteksi SSD MobileNetV2 (IOU untuk NMS = 0,2)

Gambar 4.24 Contoh deteksi YOLOv3 (IOU untuk NMS = 0,6)

Gambar 4.25 Contoh deteksi YOLOv3 (IOU untuk NMS = 0,4)

Gambar 4.26 Contoh deteksi YOLOv3 (IOU untuk NMS = 0,2)

V SIMPULAN DAN SARAN

5.1 Simpulan

Nilai mAP berbanding lurus dengan waktu komputasi (*inference time*) algoritme, yakni semakin tinggi nilai mAP maka semakin lama waktu komputasi yang dibutuhkan. Algoritme yang dapat mengekstrak fitur lebih baik maka menghasilkan kinerja deteksi yang lebih baik untuk area deteksi yang kompleks dan padat. Faster R-CNN memiliki nilai mAP tertinggi yaitu ~49% dengan waktu komputasi sekitar 2,5 detik. Sehingga, Faster R-CNN dapat digunakan untuk *object detection task* pada data citra dengan kebutuhan akurasi yang tinggi namun dapat mengabaikan kecepatan komputasi.

Selain memiliki *inference time* yang cukup cepat (2 detik) dan penggunaan sumber daya yang paling kecil (CPU 48,5% dan RSS 551%), SSD MobileNetV2 menghasilkan mAP yang dapat dipertimbangkan (33,16%) sehingga dalam kasus ini dapat direkomendasikan untuk proses *object detection* secara *realtime*. Adapun YOLOv3, meskipun memiliki waktu komputasi tercepat (0,5 detik), tetapi memiliki nilai mAP yang sangat rendah (di bawah 20%) untuk dapat mendeteksi objek daun tanaman melon beserta kelasnya. Sehingga, YOLOv3 tidak direkomendasikan untuk digunakan karena akan membuat lebih banyak kesalahan deteksi.

5.2 Saran

Penelitian selanjutnya dapat berkonsentrasi pada deteksi jenis kelainan yang lebih spesifik pada daun tanaman melon. Selain itu, proses anotasi data dapat melibatkan pakar sehingga meminimalkan terjadi kesalahan saat memberikan *bounding box* dan label kelas.

Untuk percobaan yang lebih baik, dapat diterapkan berbagai teknik pelatihan model untuk memperbaiki kinerja algoritme. Misalnya, dengan memperbanyak jumlah dataset, mencoba lebih banyak teknik augmentasi data, mengubah ukuran *input*, menggunakan *feature extractor* yang berbeda, atau melakukan *tuning* parameter dan teknik pelatihan lainnya. Adapun untuk meningkatkan kinerja algoritme baik dari aspek akurasi maupun beban komputasi, dapat dilakukan dengan memodifikasi jaringan algoritme disesuaikan dengan kebutuhan dataset atau perangkat yang digunakan.

DAFTAR PUSTAKA

Alganci U, Soydas M, Sertel E. 2020. Comparative research on deep learning approaches for airplane detection from very high-resolution satellite images. *Remote Sensing*. 12(3):458. doi: 10.3390/rs12030458.

Ampatzidis Y, Bellis LD, Luvisi A. 2017. iPathology: robotic applications and management of plants and plant diseases. *Sustainability*. 9:1-14. doi: 10.3390/su9061010.

Archana M, Geetha MK. 2015. Object detection and tracking based on trajectory in broadcast tennis video. *Procedia Computer Science*. 58:225-232. doi: 10.1016/j.procs.2015.08.060.

Arsenovic M, Karanovic M, Sladojevic S, Anderla A, Stefanovic D. 2019. Solving current limitations of deep learning based approaches for plant disease detection. *Symmetry*. 11(939):1-21. doi: 10.3390/sym11070939.

Aydin I, Othman NA. 2017. A new IoT combined face detection of people by using computer vision for security application. *International Artificial Intelligence and Data Processing Symposium (IDAP)*. doi: 10.1109/IDAP.2017.8090171.

Balliu A, Sallaku Glenda. 2017. Early production of melon, watermelon and squashes in low tunnels. Di dalam: Baudoin W, Nersisyan A, Shailov A, Hodder A, Gutierrez D, editor. Edisi 230(5). *Good Agricultural Practices for greenhouse vegetable production in the South East European countries*. Food and Agriculture Organization of the United Nations. hlm 341-351.

Dai J, Li Y, He K, Sun J. 2016. R-FCN: Object detection via region-based fully convolutional networks [Internet]. [diunduh 2020 Mei 25]. Tersedia dari: https://arxiv.org/pdf/1605.06409.

Elgendi M, Nasir MU, Tang Q, Fletcher RR, Howard N, Menon C, Ward R, Parker W, Nicolaou S. 2020. The performance of deep neural networks in differentiating chest x-rays of COVID-19 patients from other bacterial and viral pneumonias. *Frontiers in Medicine*. doi: 10.3389/fmed.2020.00550.

Foley D, O'Reilly R. 2018. An evaluation of convolutional neural network models for object detection in images on low-end devices. Di dalam: *Proceedings for the 26th Irish Conference on Artificial Intelligence and COgnitive Science* [Internet]. Trinity College, Dublin, Dublin, Ireland. [diunduh 2020 Okt 29]. Tersedia pada: http://ceur-ws.org/vol-2259/aics_32.pdf.

Girshick R. 2015. Fast R-CNN. Di dalam: *Proceedings of the IEEE International Conference on Computer Vision*. 1440–1448. doi: 10.1109/ICCV.2015.169.

Girshick R, Donahue J, Darrell T, Malik J. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. Di dalam: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 580–587. doi: 10.1109/CVPR.2014.81.

Godil A, Bostelman R, Shackleford W, Hong T, Shneier M. 2014. Performance metrics for evaluating object and human detection and tracking systems. NISTIR 7972, National Institute of Standards and Technology, Gaithersburg, MD, USA. doi: 10.6028/NIST.IR.7972.

He K, Zhang X, Ren S, Sun J. 2015. Deep residual learning for image recognition [Internet]. [diunduh 2020 Okt 22]. Tersedia dari: https://arxiv.org/pdf/1512.03385.

He Y, Zeng H, Fan Y, Ji S, Wu J. 2019. Application of deep learning in integrated pest management: a real-time system for detection and diagnosis of oilseed rape pests. *Mobile Information System*. 2019. doi: 10.1155/2019/4570808.

Hu L, Ni Q. 2018. IoT-driven automated object detection algorithm for urban surveillance systems in smart cities. *IEEE Internet of Things Journal*. 5(2):747-754. doi: 10.1109/JIOT.2017.2705560.

Huang YQ, Zheng JC, Sun SD, Yang CF, Liu J. 2020. Optimized YOLOv3 algorithm and its application in traffic flow detections. *Applied Sciences*. 10(9):3079. doi: 10.3390/app10093079.

[IPB] IPB University. 2019. IPB inaugurates smart urban farming laboratory [Internet]. Tersedia dari: https://ipb.ac.id/news/index/2019/06/ipb-inaugurates-smart-urban-farming-laboratory/21465b1a2c18b484632aa177eb02dbff.

Javier R. 2017. Object detection with deep learning: the definitive guide [Internet]. Tersedia dari: https://tryolabs.com/blog/2017/08/30/object-detection-an-overview-in-the-age-of-deep-learning/.

Kerrisk M. 2020. Pidstat(1) – Linux manual page [Internet]. [diacu 2020 Okt 29]. Tersedia pada: https://man7.org/linux/man-pages/man1/pidstat.1.html.

Khirade SD, Patil AB. 2015. Plant disease detection using image processing. Di dalam: *Proceedings of the 2015 International Conference on Computing Communication Control and Automation*. hlm 768-771. doi: 10.1109/ICCUBEA.2015.153.

Lei J, Gao C, Hu J, Gao C, Sang N. 2019. Orientation adaptive YOLOv3 for object detection in remote sensing images. Di dalam: Lin Z, Wang L, Yang J, Shi G, Tan T, Zheng N, Chen X, Zhang Y, editor. *Pattern Recognition and Computer Vision*. hlm 586-597. doi: 10.1007/978-3-030-31654-9_50.

Li J, Gu J, Huang Z, Wen J. 2019. Application research of improved YOLO V3 algorithm in PCB electronic component detection. *Applied Sciences*. 9(18):3750. doi: 10.3390/app9183750.

Li M, Zhang Z, Lei L, Wang X, Guo X. 2020. Agricultural greenhouses detection in high-resolution satellite images based on convolutional neural networks: Comparison of Faster R-CNN, YOLO v3 and SSD. *Sensors*. 20(17):1-4. doi: 10.3390/s20174938.

Liu W, Anguelov D, Erhan D, Szegedy C, Reed S, Fu CY, Berg AC. 2016. SSD: Single shot multibox detector [Internet]. doi: 10.1007/978-3-319-46448-0_2. [diunduh 2020 Mei 21]. Tersedia dari: http://arxiv.org/pdf/1512.02325.

Lu X, Li Q, Li B, Yan J. MimicDet: bridging the gap between one-stage and two-stage object detection [Internet]. [diunduh 2020 Okt 29]. Tersedia pada: https://arxiv.org/pdf/2009.11528.

Mansoub SK, Abri R, Yarıcı, AH. 2019. Concurrent real-time object detection on multiple live streams using optimization CPU and GPU resources in YOLOv3. Di dalam: *SIGNAL 2019: The Fourth International Conference on Advances in Signal, Image, and Video Processing*.

Mao QC, Sun HM, Liu YB, Jia RS. 2019. Mini-YOLOv3: Real-Time Object Detector for Embedded Applications. *IEEE Access*. 7:133529–133538. doi: 10.1109/ACCESS.2019.2941547.

Mohanty SP, David PH, Marcel S. 2016. Using deep learning for image-based plant disease detection. *Frontiers in Plant Science*. 7(1419):1-10. doi: 10.3389/fpls.2016.01419.

Nerson H. 2008. Mineral nutrition of cucurbit crops. *Dynamic Soil, Dynamic Plant*. 2(1):23-32.

Nguyen K, Huynh NT, Nguyen PC, Nguyen KD, Vo ND, Nguyen TV. 2020. Detecting objects from space: an evaluation of deep-learning modern approaches. *Electronics*. 9(583). doi: 10.3390/electronics9040583.

Nguyen ND, Do T, Ngo TD, Le DD. 2020. An evaluation of deep learning methods for small object detection. *Journal of Electrical and Computer Engineering*. 2020. doi: 10.1155/2020/3189691.

Panchal P, Prajapati G, Patel S, Shah H, Nasriwala J. 2015. A review on object detection and tracking methods. *International Journal for Research in Emerging Science and Technology*. 2(1):7-12.

Pineda M, Pérez-Bueno ML, Barón M. 2018. Detection of bacterial infection in melon plants by classification methods based on imaging data. *Frontiers in Plant Science*. 9(164):1-10. doi: 10.3389/fpls.2018.00164.

Prajapati D, Galiyawala HJ. 2015. A review on moving object detection and tracking. *International Journal of Computer Application*. 5(3):168-175.

Ramcharan A, McCloskey P, Baranowski K, Mbilinyi N, Mrisho L, Ndalahwa M, Legg J, Hughes DP. 2019. A mobile-based deep learning model for cassava disease diagnosis. *Frontiers in Plant Science*. 10(272):1-8. doi: 10.3389/fpls.2019.00272.

Redmon J, Divvala S, Girshick R, Farhadi A. 2016. You only look once: Unified, real-time object detection [Internet]. doi:10.1109/CVPR.2016.91. [diunduh 2020 Mei 27]. Tersedia dari: http://arxiv.org/pdf/1506.02640.

Redmon J, Farhadi A. 2017. YOLO9000: Better, faster, stronger [Internet]. doi:10.1109/CVPR.2017.690. [diunduh 2020 Mei 27]. Tersedia dari: http://arxiv.org/pdf/1612.08242.

Redmon J, Farhadi A. 2018. YOLOv3: An Incremental Improvement [Internet]. [diunduh 2020 Mei 27]. Tersedia dari: http://arxiv.org/pdf/1804.02767.

Ren S, He K, Girshick R, Sun J. 2017. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Trans. Pattern Anal. Mach. Intell*. 39(6):1137–1149. doi:10.1109/TPAMI.2016.2577031.

Rezatofighi H, Tsoi N, Gwak J, Sadeghian A, Reid I, Savarese S. 2019. Generalized intersection over union: A metric and a loss for bounding box regression [Internet]. doi:10.1109/CVPR.2019.00075. [diunduh 2020 Mei 27]. Tersedia dari: http://arxiv.org/pdf/1902.09630.

Sa I, Ge Z, Dayoub F, Upcroft B, Perez T, McCool C. 2016. DepFruits: a fruit detection system using deep neural networks. *Sensors*. 16(1222):1-23. doi: 10.3390/s16081222.

Sandler M, Howard A, Zhu M, Zhmoginov A, Chen LC. 2019. MobileNetV2: inverted residuals and linear bottlenecks [Internet]. [diunduh 2020 Okt 20]. Tersedia dari: https://arxiv.org/pdf/1801.04381.

Sermanet P, Eigen D, Zhang X, Mathieu M, Fergus R, LeCun Y. OverFeat: Integrated recognition, localization and detection using convolutional networks [Internet]. Tersedia dari: https://arxiv.org/pdf/1312.6229v4.

Srivastava A, Nguyen D, Aggarwal S, Luckow A, Duffy E, Kennedy K, Ziolkowski M, Apon A. 2019. Performance and Memory Trade-offs of Deep Learning Object Detection in Fast Streaming High-Definition Images. Di dalam: *Proceedings - 2018 IEEE International Conference on Big Data*. hlm. 3915–3924.

Sultana F, Sufian A, Dutta P. 2019. A review of object detection models based on convolutional neural network. *2nd International Conference on Communication, Devices and Computing* [Internet]. [diunduh 2020 Mei 25]. Tersedia dari: https://arxiv.org/pdf/1905.01614.

Trigueros DS, Meng L, Hartnett M. 2018. Face recognition: from traditional to deep learning methods [Internet]. Tersedia dari: https://arxiv.org/pdf/1811.00116.

Wei L, Anguelov D, Erhan D, Szegedy C, Reed S, Fu CY, Berg AC. 2016. SSD: Single Shot MultiBox Detector [Internet]. [diacu 2020 Mei 25]. Tersedia dari: https://arxiv.org/pdf/1512.02325.

Yadav P, Kumari M. 2018. Object Recognition Using Deep Learning in IoT Applications. *3rd International Conference on Internet of Things and Connected Technologies*. doi: 10.2139/ssrn.3168605.

Zhong Y, Gao J, Lei Q, Zhou Y. 2018. A vision-based counting and recognition system for flying insects in intelligent agriculture. *Sensors*. 18(1489):1-19. doi: 10.3390/s18051489.

Zoph B, Cubuk ED, Ghiasi G, Lin TY, Shlens J, Le QV. 2019. Learning data augmentation strategies for object detection [Internet]. [diunduh 2020 Mei 25]. Tersedia dari: https://arxiv.org/pdf/1906.11172v1.

RIWAYAT HIDUP

Penulis lahir di Kuningan pada 24 Mei 1993 dari ayah Drs. H. Udin Shofiyudin (Alm.) dan Dra. Hj. Yayah Juhaeriyah. Penulis merupakan anak ketiga dari tiga bersaudara.

Pada tahun 2011 penulis menyelesaikan sekolah di SMK K.H.A. Wahab Muhsin, Tasikmalaya, kemudian melanjutkan pendidikannya di Program Studi Statistika Universitas Islam Indonesia (UII), Yogyakarta. Selama menempuh pendidikan sarjana (S1), penulis meraih beasiswa unggulan *full study* dari Pondok Pesantren Universitas Islam Indonesia. Setelah lulus dari UII pada tahun 2015, penulis melaksanakan program pengabdian di kampus tersebut dan bekerja di sana. Pada tahun 2018, penulis menerbitkan sebuah buku yang berjudul "Islam dalam Uji Hipotesis". Di tahun yang sama, penulis mulai menempuh pendidikan magister (S2) di Program Studi Magister Ilmu Komputer Institut Pertanian Bogor (IPB).